Testing de Software
Trabajo Práctico Obligatorio 1

La resolución de este trabajo práctico será grupal. Cada grupo debe tener dos integrantes. La fecha límite para la presentación es el día 20/4 a las 12hs. El proyecto cuenta con una instancia de defensa oral individual, que se realizará el 20/4 a partir de las 16hs. Para la regularidad de la materia deben aprobarse los tres trabajos prácticos obligatorios con nota mínima de 5 puntos.

En la evaluación se tendrá en cuenta la comprensión de los conceptos teóricos y la capacidad del estudiante de aplicarlos en la práctica. Para las soluciones se tendrán en cuenta criterios estándar de calidad de los tests: explorar minuciosamente los comportamientos del software, contar con aserciones apropiadas ("completitud" de las aserciones en property-based testing), legibilidad de los tests, calidad de los generadores automáticos, etc…
Importante: Se restarán hasta 5 puntos (sobre un total de 10) por compartir soluciones entre distintos grupos.

Ejercicio 0. Descargue el código provisto en el repositorio git a continuación:

https://github.com/pponzio/testing-unrc-2022-TP1

El repositorio contiene el código a testear en este trabajo práctico. Se incluyen scripts gradle para compilar automáticamente el código y ejecutar los tests (ej. ejecutar *gradle test* en línea de comandos). Se recomienda utilizar algún IDE (ej. Eclipse) para facilitar la codificación, la ejecución de tests y el debugging.

Ejercicio 1. Escriba propiedades para todos los programas con defectos del ejercicio 1 de la práctica 1 (además de `findLast`, ya resuelto en la teoría). Los programas con defectos se proveen en la clase `testing.tp1.ejercicio1.Ejercicio1.java`. Use AssertJ para definir las propiedades. Intente que las propiedades capturen con la mayor precisión posible la especificación de cada programa. Usando los valores definidos para los generadores de los tests parametrizados del ejercicio 1 de la práctica anterior, verifique que las propiedades que escribió son capaces de revelar los defectos.

Ejercicio 2. Modifique las propiedades propuestas como solución del ejercicio 1 para usar jqwik como generador automático de entradas. Emplee los generadores apropiados para que jqwik sea capaz de revelar las fallas en todos los casos. Modifique o agregue propiedades en caso de que sea necesario para detectar todas las fallas.

Ejercicio 3. Provea tests parametrizados (con entradas en formato CSV) para los métodos de la clase `testing.tp1.ejercicio3.BoundedQueue.java`. Asegúrese de incluir tests de unidad y de módulo, que ejerciten todas las rutinas de la clase.

Ejercicio 4. Escriba propiedades y generadores de valores para los métodos de la clase `testing.tp1.ejercicio3.BoundedQueue.java`. Implemente iteradores para `BoundedQueue` de manera que sea posible usar las aserciones de iterables de AssertJ en las propiedades. Defina propiedades que capturen las especificaciones de los métodos con la mayor precisión posible. Asegúrese de incluir tests de unidad y de módulo, que ejerciten todas las rutinas de la clase. ¿Qué diferencias hay entre los tests parametrizados del ejercicio anterior y los tests basados en propiedades de este ejercicio?

Ejercicio 5. Se proveen dos clases, `testing.tp1.ejercicio5.Point` y `testing.tp1.ejercicio5.ColorPoint`, con métodos `equals` defectuosos. Averigüe qué propiedades debe satisfacer el método `equals` en Java e implemente estas propiedades como tests JUnit. Defina generadores de valores jqwik para estas propiedades que sean capaces de revelar las fallas en las implementaciones de `equals`.