Relatório Static Testing : Jdotxt

Ferramenta utilizada:

- SpotBugs

Static Testing:

- Static Testing é uma técnica utilizada para testar o código produzido sem necessidade de execução. Este tipo de testes são feitos em estados iniciais do desenvolvimento, devido a ser mais fácil identificar este tipo de erros.
- Há 2 tipos de testes estáticos:
    - Feitos manualmente em que é feita análise do código, chamadas **reviews**
    - Feitos automaticamente recorrendo a ferramentas próprias para esse efeito, como por exemplo SonarLint, SpotBugs, PMD, etc...
- Este tipo de testes são utilizados para eliminar erros e ambiguidades, como requisitos, casos de teste, etc.
- Alguns dos defeitos que poderão ser encontrados incluem:
    - Variáveis sem valores definidos;
    - Unreachable / dead Code;
    - Violação de “boas práticas” de programação;
    - entre outros.
- Assim, static testing é importante de modo a evitar custos (tanto temporais como monetários) futuros, visto que muitos erros provém de erros / má programação nos estados iniciais do projeto, devido à natureza incremental de um projeto de programação.

Bugs Selecionados para resolução:

- *ThresholdDateParser.pare(String,Pattern), Normal Confidence:*
    - Este erro era demonstrado pelo SpotBugs devido ao facto que funções estáticas que não são Thread-Safe eram utilizadas pelo projeto, que utilizava a interface runnable.
        - A resolução passou por garantir sincronização quando estes métodos são utilizados, utilizando a keyword **synchronized.**

- *Util.createParentDirectory(File), Normal Confidence*:
    - Erro demonstrado pelo spot bugs devido à possibilidade de um Null Pointer que terminaria inesperadamente a execução do programa.
        - A resolução passou por colocar uma verificação no caso do pointer para a diretoria em questão ser null, ignorar as operações sobre ela.
- *Jdotxt.onWindow(), High Confidence:*
    - Erro demonstrado pelo spot bugs devido ao facto que a cláusula try/catch seria sempre ignorada, devido ao facto que nenhuma função utilizada tinha na sua documentação “throws Exception”.
        - A resolução passou por retirar esta cláusula Try/Catch, retirando também *catch(Error e)*, visto que segundo a API da classe Error, estes representam comportamento anormal que não deve ser tratado.

- *Classe JdotxtGUI, Normal Confidence:*
    - Erro demonstrado devido a constantes estáticas da classe não estarem descritas com a keyword **final** de modo a não poderem ser alteradas em tempo de execução.
        - A resolução passou por adicionar esta keyword, de modo a não ser possível alterar constantes que iriam alterar o comportamento do programa.
- *JdotxtWelcomeDialog.initGUI() - Default case missing, Normal Confidence:*
    - Erro demonstrado devido à existência da keyword **switch** sem **default**. Caso existisse alguma situação em que comportamento inesperado do programa alterasse o caminho de execução, seria possível todo o código dentro deste scope fosse ignorado.
        - A resolução passou por adicionar um caminho de execução que passou por adicionar a keyword **default.**

Descrição do projeto e como está organizado

O projeto jdotxt fornecido é um programa Desktop que permite a organização da lista de tarefas de um utilizador. Para isso, lança uma aplicação GUI que permite ao utilizador comunicar-se com uma interface mais usável.

Olhando para os módulos que compõem este projeto, aparente estar organizado em camadas:

- A primeira camada (de apresentação) que é responsável pela interface para o mundo através de oferta de serviços e/ou apresentação de informação ao utilizador
    - Packages encarregues do lançamento do GUI, como o *com.chschmid.jdotxt.gui*
- A camada de negócio que é responsável por todo o processamento que permite concretizar os pedidos de serviços recebidos na apresentação e calcular eventuais respostas, assim como tratamento de excepções.
    - Packages encarregues do tratamento de serviços, como o *com.chschmid.jdotxt* e *com.chschmid.jdotxt.util*
- A camada de dados que é responsável pela comunicação com base de dados, sistemas de mensagens, gestores de transações ou outros sistemas
    - Neste programa, os dados são persistidos guardando-os num ficheiro local que a aplicação acede quando lançada

Também há um package *com.todotxt.todotxttouch.task* que tem alguns testes unitários a alguns módulos do programa.

Descrição da ferramenta e a sua configuração

A ferramenta usada para realizar testes estáticos foi o SpotBugs. Esta ferramenta é usada para detectar bugs em programas Java. Destaca-se de outras devido à quantidade enorme de diferentes padrões que encontra em programas, assim como mostra um grau de confiança, para um determinado bug observado, que simplifica a análise de falsos positivos.

Para apurar resultados fiáveis, que demonstrem erros relevantes no âmbito do projeto, tivemos de restringir algumas configurações do SpotBugs.

Por exemplo, subimos o grau mínimo de confiança, a ser detectado, para Médio e também o rank mínimo para 16 de modo a encontrar bugs de rank “Of Concern”. Assim, conseguimos filtrar padrões que não tenham importância ou que não sejam prejudiciais para o projeto.

Também desativamos a procura por bugs de categoria Security, Internationalization e Performance, visto que o projeto não oferece funcionalidades relacionadas com idioma, não interessa se os inputs podem criar vulnerabilidades exploráveis ou se é ineficiente na sua execução. A categoria Experimental também foi desativada visto que não temos interesse em saber se há bugs com padrões ainda não identificados.

Por outro lado, adotamos a categoria Multithreaded correctness visto que este projeto recorre a uma interface Runnable para executar a Interface do programa e, por isso, erros de sincronização de threads podem ocorrer.

Descrição do Report produzido pela ferramenta SpotBugs:

- O SpotBugs produz um relatório num formato .xml para o programador exportar. Acaba por ser algo apreciado, mas para a situação em questão, é difícil utilizar o mesmo, devido ao seu formato.
- No entanto, este report acaba por conter um conjunto de informação importante, como:
    - A prioridade com qual o erro deve ser corrigido;
    - A Categoria do erro;
    - O Ficheiro/Linha em que se encontra;
- Todos estes erros estão categorizados e separados pelo nome “BugInstance”.