CSIE5142/CSIE4302 Software Engineering

105599004 林慧琪 Homework # 4

Due on 01/03/2018

1. (10%) Describe the 5 key activities in an object-oriented design process.

(1) Define the context and modes of use of the system:

定義清楚系統內使用的語境(函式名稱)與使用模式

(2) Design the system architecture:

設計系統架構

(3) Identify the principal system objects

識別主體系統物件

(4) Develop design models

開發設計模組

(5) Specify object interfaces

制定物件介面

2. (20%) Consider software evolution?

(a) (8%) What are the differences between refactor and reengineering?

Reengineering: Takes place after a system has been maintained for some time and maintenance costs are increasing. Use automated tools to process and reengineer a legacy system to create a new system that is more maintainable

Refactoring: A continuous process of improvement throughout the development and evolution process. It is intended to avoid the structure and code degradation that increases the costs and difficulties of maintaining a system

(b) (6%) Describe the tree types of software maintenances?

(1) Corrective maintenance

識別、隔離、修復故障，使設備回復至運行狀態的維護

(2) Adaptive maintenance

交付成品後再對成品進行修改，以應對在不同的執行環境下使用成品

(3) Perfective maintenance

交付成品後對成品持續維護、更新、修正，提成品性能或是可維護性

(c) (6%) What are bad smells (or code smells)? How will you relate bad smells and preventative maintenance?

Bad smells指的是程式碼中設計不良，或是後續難以維護的部分，以下列出幾個常見的程式碼Bad smells:

(1) 重複的程式碼(Duplicated Code)

一個以上的地方出現相同的程式碼結構，導致浪費記憶空間與增加程式修改的難度，改善方式為將相同的部分整合成一個function，需要時在去呼叫，或是套用Extract Method、Pull Up Field、Form Template Method、Substitute Algorithm、Extract Class來移除duplicated code

(2) 過長函式(Long Method)

指一個函式太過冗長，導致程式的可讀性變差，函式的功能也變得過於複雜，改善方式為套用Extract Method、Replace Temp with Query、Introduce Parameter Object、Preserve Whole Object、Replace Method with Method Object、與Decompose Conditional

(3) switch驚悚現身(Switch Statements)

本質上來說switch的問題就在於重複，常看到相同的switch出現在不同的地方，如要添加一個case就要找到所有的switch，增加程式的修改難度，改善方式就是使用多型(polymorphism)去達成同樣效果，或是套用Move Method和Move Field

(4) 資料泥團(Data Clumps)

Data clumps 是指「總是一起出現的資料」，例如出現在不同類別或是不同函數參數列的資料，像是資料庫connection string所需使用的多個參數（id、password、ip、port number、database name、database arguments等），在不同的地方重複宣告多個相同的資料組，可以藉由套用Extract Class、Introduce Parameter Object、Preserve Whole Object來進行Data clumps的改善。

(5) Speculative generality (過度臆測未來「不必要」的擴充性)

只在進行程式設計的時候，預留了太多未來可能會用到的擴充點，導致造成過多事前設計，變成資源浪費與增加日後維護的困難，可以套用Collapse Hierarchy、Inline Class、Remove Parameter和Rename Method等方法進行改善。

3. (20%) Consider a program that takes a numerical score (ranged from 0 to 100)

and transfers the score to a letter grade A(score>=90), B(80<=score<90), C(70<=score<80), D(60<=score<70), or F(score<60); otherwise X (score<0 or score > 100).

(a) (10%) Apply the equivalence partitioning testing technique to design test cases for testing the program.

有效類別資料:介於0~100之間的數字

無效類別資料:小於0,大於100,非數字,空值

|  |  |  |
| --- | --- | --- |
| No. | 輸入成績 | 預期結果 |
| 1 | 14 | F |
| 2 | -3 | X |
| 3 | 0 | F |
| 4 | 66 | D |
| 5 | 78 | C |
| 6 | 87 | B |
| 7 | 99 | A |
| 8 | 120 | X |

(b) (10%) Based on your answers in (a), design additional test cases by applying

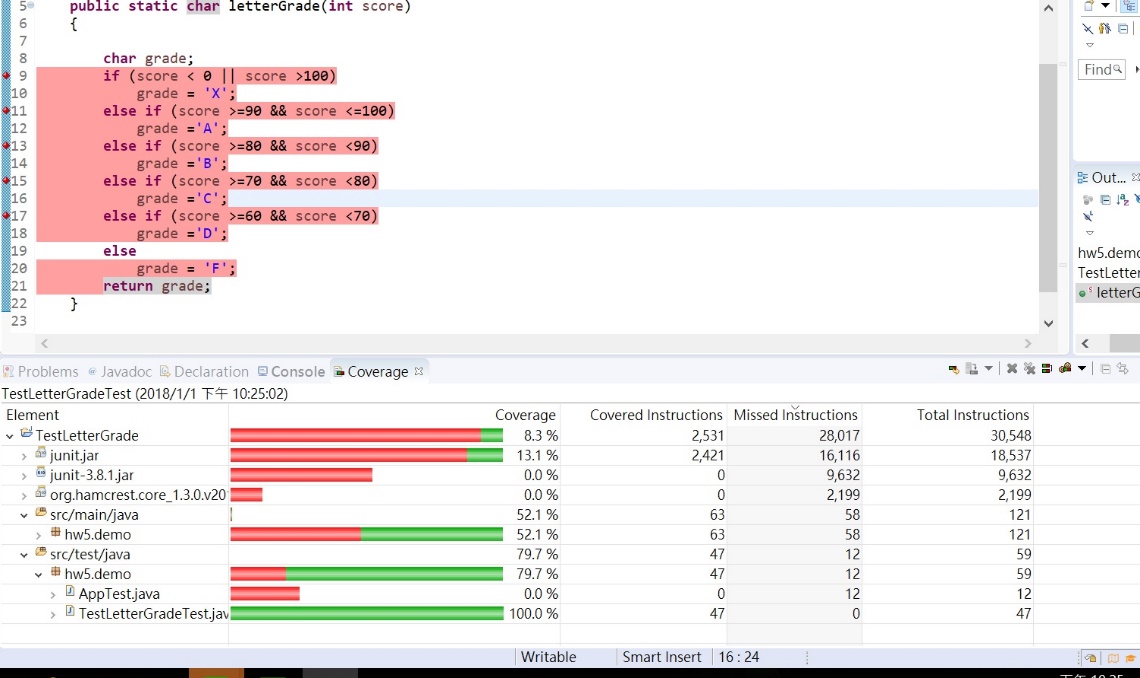
the boundary value analysis testing technique.

|  |  |  |
| --- | --- | --- |
| No. | 輸入成績 | 預期結果 |
| 1 | -1 | X |
| 2 | 0 | F |
| 3 | 1 | F |
| 4 | 99 | A |
| 5 | 100 | A |
| 6 | 101 | X |
| 7 | 65 | D |

4. (30%) Consider the following program.

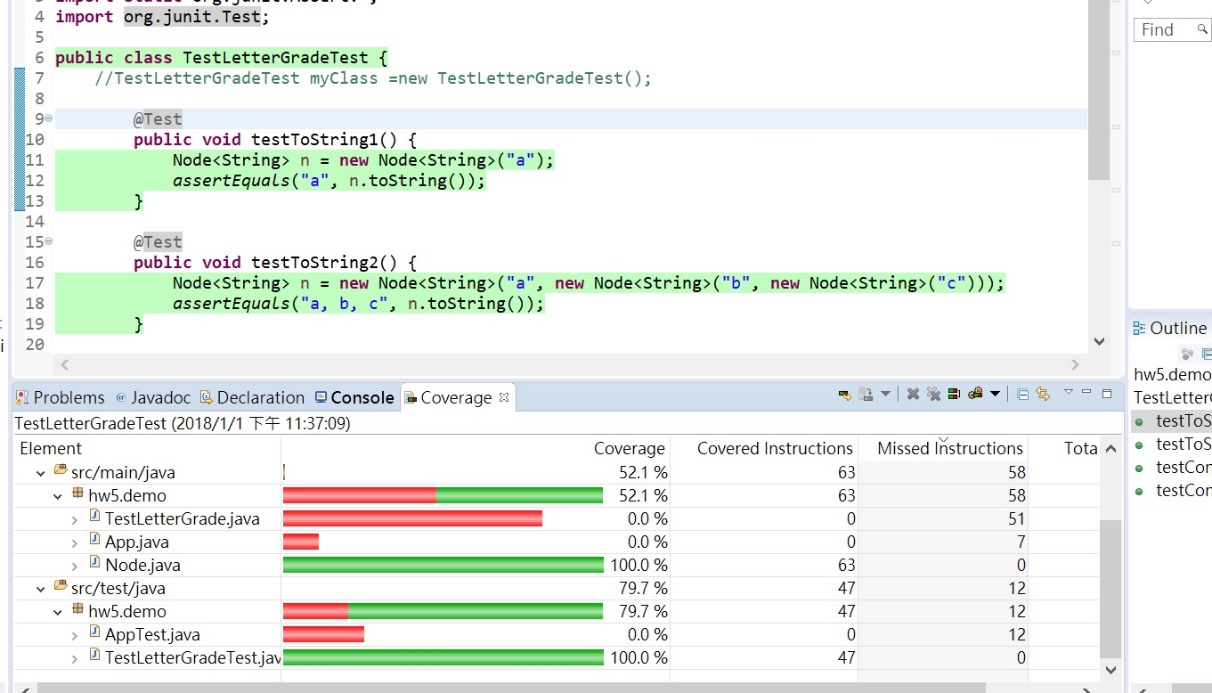
(a) (15%) Implement your test cases in problem 4(a) using JUnit. Show the

JUnit source code of your test cases and the screen snapshots of the

execution results of the test cases (including code coverage).

(b) (15%) Implement your test cases in problem 4(b) using JUnit. Show the

JUnit source code of your test cases and the screen snapshots of the execution esults of the test cases (including code coverage)



5. (20%) Illustrate the application of the JUnit and configuration management (CM) tools, such as subversion or Git (or GitHub), in software development. Note that you may integrate your IDE tool with your chosen CM tool, and you also need to create your own repository using the chosen CM tool.

(a) (10%) Show the screen snapshots for using the CM tool to check in the source code of letterGrade.java and then check out the code to add a main( ) function so that the program can be executed and tested in console manually. After the manual testing is completed and the program is correct, commit the source code to the repository.

(b) (10%) Show the screen snapshots for using the CM and JUnit tools to check out your source code of letterGrade.java committed in 5(a) and add JUnit test cases to test the program automatically. After all the test cases are pass and the statement coverage is 100%, commit you source code and test cases to the repository