軟體工程 Homework #4

Part 1.

Define the context and modes of use of the system:

了解軟體和外在環境之間的關係，來決定如何提供系統功能，以及系統與外在環境的溝通，並且了解情境來決定系統的界線，以此來決定各系統所要實作的功能。

Context model 是一種架構模型，展示了環境中的不同系統，以區塊圖(block diagram)呈現。

Interaction model 是一種動態模型，展示了系統與環境是如何互動，以使用案例模型呈現。

Design the system architecture:

確認系統的主要構件，並且用 architectural pattern，如 layered model 或是 client-server model 來組織構件。

Identify the principal system objects:

想要確認系統中有那些構件通常會需要經驗和相關的領域知識，確認系統中存在之物件通常是一件連續性的事情，非正式的系統描述文件等等可以做為確認物件、物件屬性、物件的行為的起手點。

Develop design models:

設計模型展示了物件、物件類別、以及這些實體之間的關係，決定設計模型以及設計模型所要求的細節，是設計階段中最重要的步驟。

Subsystem models 顯示了將物件連貫成子系統的分群邏輯

Sequence models 顯示了物件互動的次序

State machine models 顯示了物件為了回應事件，會如何去改變自身狀態

Specify object interfaces:

物件的介面必須定義清楚，才能使得物件和其他構件可以並行設計，並且要注意不要將介面的過多細節透漏給物件或是一群物件，設計介面時也應該要避免將底層資料表示法暴露。

Part 2.

Coupling:

是程式中的模組彼此相依程度的等級，等級越高表示模組之間干擾彼此或依存彼此的程度越高，其中又分 interface coupling 和 implementation coupling。

Cohesion:

模組內成員彼此之間的內部可靠度等級，一個高 cohesion 的類別，越能根據他的介面就了解該類別所負責的行為。

Part 3.

Version management:

持續追蹤各軟體構件的不同版本，並且促進程式撰寫人員合作開發。

System integration:

定義構件的哪一些版本會產生對應的系統版本，並且利用這些資訊來自動建置系統。

Problem tracking:

使用者可以回報錯誤和問題，並且工程師可以看到處理問題的負責人和處理、解決時間。

Release management:

當系統釋出新版本時，會需要管理的事項包含：計畫新版本的功能、組織軟體版本。

Part 4.

(a)

Refactor 重構，最主要的目的是要在不改變軟體可察覺行為的前提下使用

重構方法調整軟體結構，並且是一個持續性的事件，其中重構的方式包含：Rename、Extract method、Move method、Extract class 等等，並且同一種方式可以作用在不同結構上，產生不同層次的設計改善，比如 Rename Method、Rename Class、Rename Variables 等等。

Reengineering 重工，是指對既存軟體系統進行調查，並重新開發的過程，其目的是重新審視現有的系統，以便進一步利用新技術來改善系統或促進現存系統的再利用。

(b)

Maintenance to repair software faults

修改系統以修正系統的缺陷，使其達到需求，也稱 Corrective maintenance。

Maintenance to adapt software to a different operating environment

修改系統使其可以在與原本作業系統不相同的作業系統上運作，也稱 Adaptive maintenance。

Maintenance to add or to modify the system's functionality

修改系統使其可以滿足新需求，也稱 Perfective maintenance。

(c)

Bad smells，程式裡的壞味道，通常指的是程式碼中一些導致邏輯隱晦不清，或者冗長且重複出現，又或者容易產生瑕疵等等的程式段落，例如：Duplicate code、Long methods、Switch (case) statements、Data clumping、Speculative generality。

Preventative maintenance 是一種預防 bad smells 的手段，可以透過 refactoring 來達成，由於重構是一種持續改善程式品質的流程，所以可以藉由重構來減緩修改程式所帶來的錯誤或瑕疵。

Part 5.

(a)

{partition of score: 0 ~ 59, <input: 0, expect output: 'F'>}
{partition of score: 0 ~ 59, <input: 59, expect output: 'F'>}

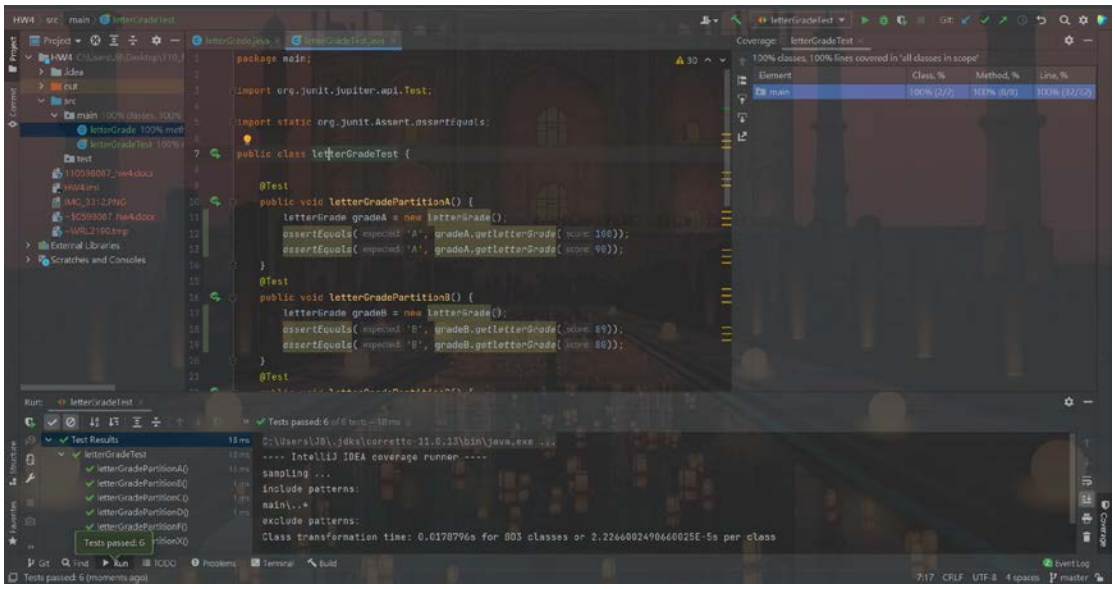
{partition of score: 60 ~ 69, <input: 60, expect output: ‘D’>}
{partition of score: 60 ~ 69, <input: 69, expect output: ‘D’>}
{partition of score: 70 ~ 79, <input: 70, expect output: ‘C’>}
{partition of score: 70 ~ 79, <input: 79, expect output: ‘C’>}
{partition of score: 80 ~ 89, <input: 80, expect output: ‘B’>}
{partition of score: 80 ~ 89, <input: 89, expect output: ‘B’>}
{partition of score: 90 ~ 100, <input: 90, expect output: ‘A’>}
{partition of score: 90 ~ 100, <input: 100, expect output: ‘A’>}
{partition of score: less than 0, <input: -1, expect output: ‘X’>}
{partition of score: greater than 100, <input: 101, expect output: ‘X’>}

(b)

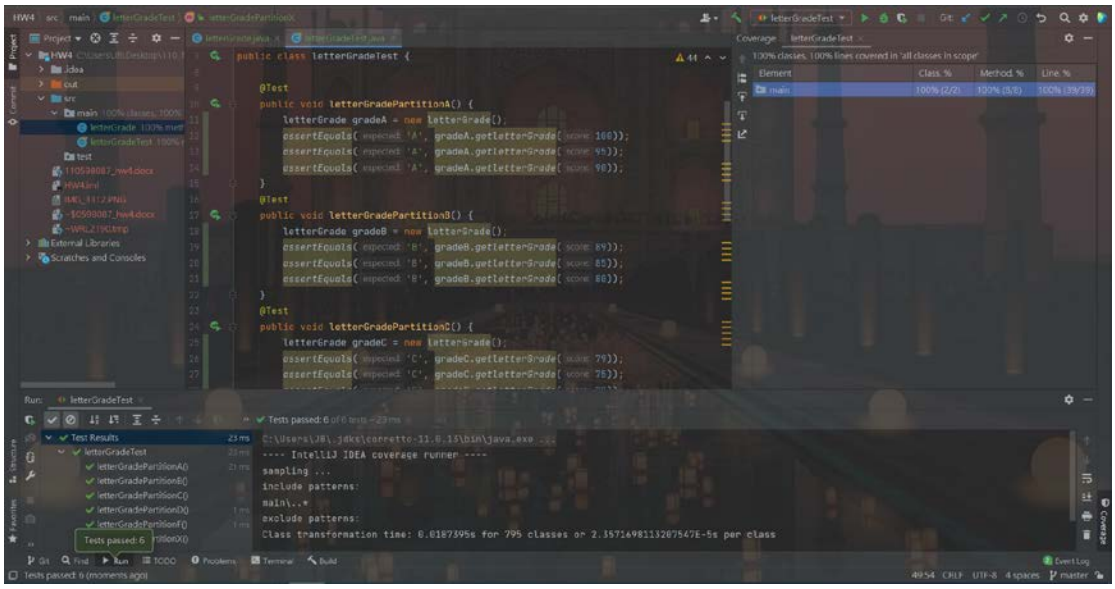
{boundary: 0 ~ 59, <input: 0, expect output: ‘F’>}
{boundary: 0 ~ 59, <input: 30, expect output: ‘F’>}
{boundary: 0 ~ 59, <input: 59, expect output: ‘F’>}
{boundary: 60 ~ 69, <input: 60, expect output: ‘D’>}
{boundary: 60 ~ 69, <input: 65, expect output: ‘D’>}
{boundary: 60 ~ 69, <input: 69, expect output: ‘D’>}
{boundary: 70 ~ 79, <input: 70, expect output: ‘C’>}
{boundary: 70 ~ 79, <input: 75, expect output: ‘C’>}
{boundary: 70 ~ 79, <input: 79, expect output: ‘C’>}
{boundary: 80 ~ 89, <input: 80, expect output: ‘B’>}
{boundary: 80 ~ 89, <input: 85, expect output: ‘B’>}
{boundary: 80 ~ 89, <input: 89, expect output: ‘B’>}
{boundary: 90 ~ 100, <input: 90, expect output: ‘A’>}
{boundary: 90 ~ 100, <input: 95, expect output: ‘A’>}
{boundary: 90 ~ 100, <input: 100, expect output: ‘A’>}
{boundary: less than 0, <input: -1, expect output: ‘X’>}
{boundary: less than 0, <input: -10, expect output: ‘X’>}
{boundary: greater than 1000, <input: 101, expect output: ‘X’>}
{boundary: greater than 1000, <input: 110, expect output: ‘X’>}

Part 6.

(a)

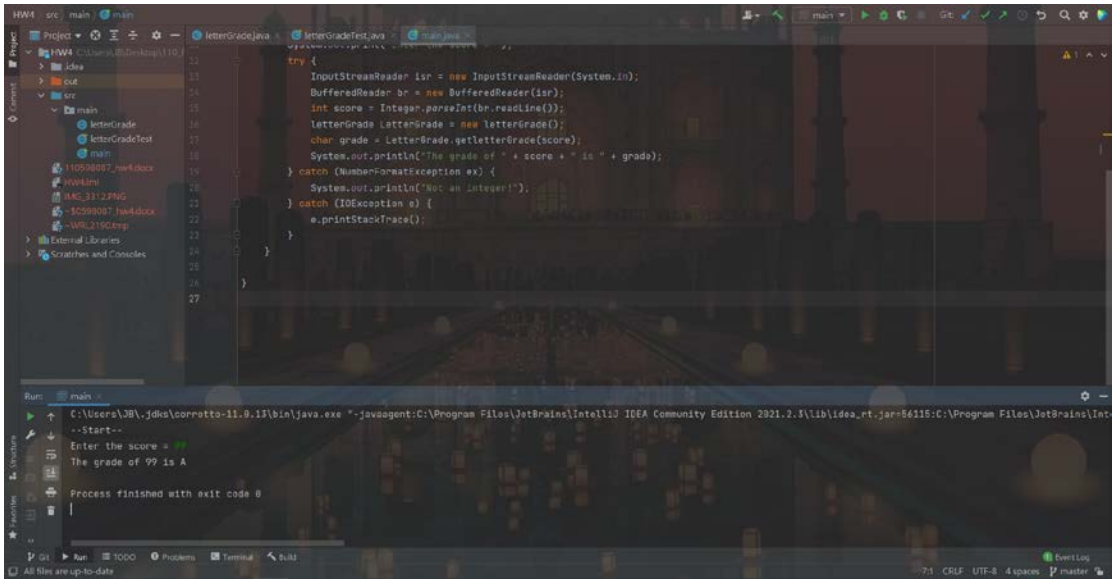

package main;
import org.junit.jupiter.api.Test;
import static org.junit.Assert.assertEquals;
public class letterGradeTest {
@Test
public void letterGradePartitionA() {
letterGrade gradeA = new letterGrade();
assertEquals(expected: 'A', gradeA.getletterGrade(score: 100));
assertEquals(expected: 'A', gradeA.getletterGrade(score: 90));
}
@Test
public void letterGradePartitionB() {
letterGrade gradeB = new letterGrade();
assertEquals(expected: 'B', gradeB.getletterGrade(score: 89));
assertEquals(expected: 'B', gradeB.getletterGrade(score: 80));
}
@Test
Tests passed: 6
---- IntelliJ IDEA coverage runner ----
sampling ...
include patterns:
main\..*
exclude patterns:
Class transformation time: 0.0178796s for 803 classes or 2.2266002490660025E-5s per class

(b)


public class letterGradeTest {
@Test
public void letterGradePartitionA() {
letterGrade gradeA = new letterGrade();
assertEquals(expected: 'A', gradeA.getletterGrade(score: 100));
assertEquals(expected: 'A', gradeA.getletterGrade(score: 95));
assertEquals(expected: 'A', gradeA.getletterGrade(score: 90));
}
@Test
public void letterGradePartitionB() {
letterGrade gradeB = new letterGrade();
assertEquals(expected: 'B', gradeB.getletterGrade(score: 89));
assertEquals(expected: 'B', gradeB.getletterGrade(score: 85));
assertEquals(expected: 'B', gradeB.getletterGrade(score: 80));
}
@Test
public void letterGradePartitionC() {
letterGrade gradeC = new letterGrade();
assertEquals(expected: 'C', gradeC.getletterGrade(score: 79));
assertEquals(expected: 'C', gradeC.getletterGrade(score: 75));
Tests passed: 6
---- IntelliJ IDEA coverage runner ----
sampling ...
include patterns:
main\..*
exclude patterns:
Class transformation time: 0.0187395s for 795 classes or 2.3571698113207547E-5s per class

Part 7.

(a)


try {
InputStreamReader isr = new InputStreamReader(System.in);
BufferedReader br = new BufferedReader(isr);
int score = Integer.parseInt(br.readLine());
LetterGrade LetterGrade = new letterGrade();
char grade = LetterGrade.getletterGrade(score);
System.out.println("The grade of " + score + " is " + grade);
} catch (NumberFormatException ex) {
System.out.println("Not an integer!");
} catch (IOException e) {
e.printStackTrace();
}
--Start--
Enter the score = 99
The grade of 99 is A
Process finished with exit code 0

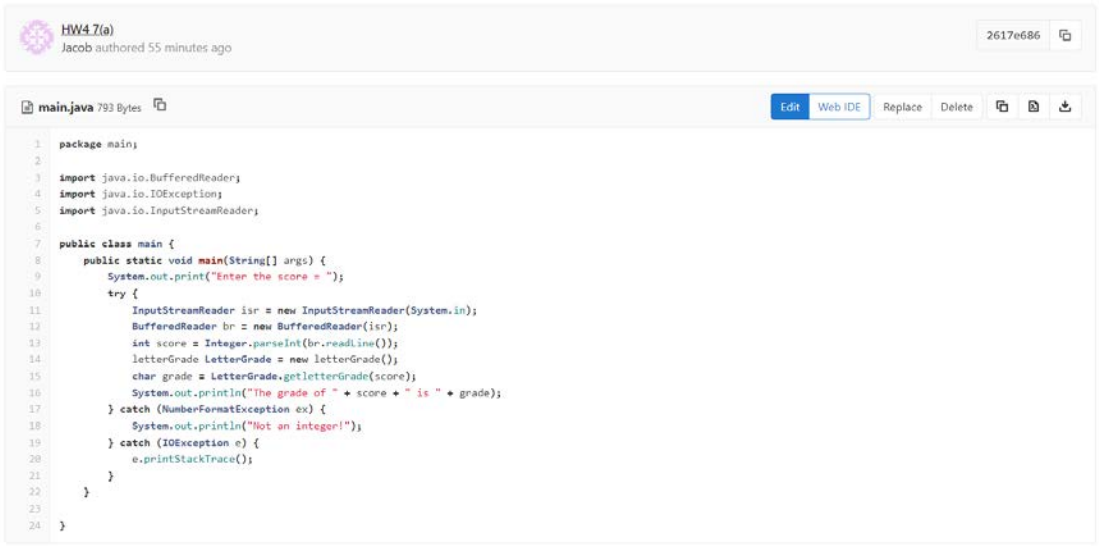
```
No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   src/main/letterGrade.java
        new file:   src/main/letterGradeTest.java
        new file:   src/main/main.java
        new file:   ../Project/SE_HW4/.idea/runConfigurations.xml
        new file:   ../Project/SE_HW4/.idea/vcs.xml

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        ../../Agile/
        ../../Embedded/
        ../../POSD/
        ../HW1/
        ../HW2/
        ../HW3/
        .idea/
        110598087_hw4.docx
        HW4.iml
        IMG_3312.PNG
        out/
        ~$0598087_hw4.docx
        ~WRL2190.tmp
        ../Project/Gan_PEP.drawio.png
        ../Project/My First Board.jpg
        ../Project/PEP_ver1.314.docx
        ../Project/Revised_srs1.2.docx
        ../Project/SDD/
        ../Project/SETeam2/
        ../Project/SE_HW4/.idea/workspace.xml
        "../Project/SE\346\234\237\344\270\255\345\240\261\345\221\212_V.pptx"
        ../teaching material/
        ../../presentation/


JB@DESKTOP-NKVBO4M MINGW64 ~/Desktop/110_first/SE/HW4 (master)
$ git commit -m "HW4 7(a)"
[master (root-commit) 2617e68] HW4 7(a)
 5 files changed, 117 insertions(+)
 create mode 100644 SE/HW4/src/main/letterGrade.java
 create mode 100644 SE/HW4/src/main/letterGradeTest.java
 create mode 100644 SE/HW4/src/main/main.java
 create mode 100644 SE/Project/SE_HW4/.idea/runConfigurations.xml
 create mode 100644 SE/Project/SE_HW4/.idea/vcs.xml

JB@DESKTOP-NKVBO4M MINGW64 ~/Desktop/110_first/SE/HW4 (master)
$ git push http://140.124.181.35:3010/110598087/git_test.git
Enumerating objects: 14, done.
Counting objects: 100% (14/14), done.
Delta compression using up to 8 threads
Compressing objects: 100% (9/9), done.
Writing objects: 100% (14/14), 1.80 KiB | 230.00 KiB/s, done.
Total 14 (delta 0), reused 0 (delta 0), pack-reused 0
To http://140.124.181.35:3010/110598087/git_test.git
 * [new branch]      master -> master
```



(b)

```java
public class letterGradeTest {

    @Test
    public void letterGradePartitionA() {
        letterGrade gradeA = new letterGrade();
        assertEquals( expected: 'A', gradeA.getletterGrade( score: 100));
        assertEquals( expected: 'A', gradeA.getletterGrade( score: 95));
        assertEquals( expected: 'A', gradeA.getletterGrade( score: 90));
    }
    @Test
    public void letterGradePartitionB() {
        letterGrade gradeB = new letterGrade();
        assertEquals( expected: 'B', gradeB.getletterGrade( score: 89));
        assertEquals( expected: 'B', gradeB.getletterGrade( score: 85));
        assertEquals( expected: 'B', gradeB.getletterGrade( score: 80));
    }
    @Test
    public void letterGradePartitionC() {
        letterGrade gradeC = new letterGrade();
        assertEquals( expected: 'C', gradeC.getletterGrade( score: 79));
        assertEquals( expected: 'C', gradeC.getletterGrade( score: 75));
        assertEquals( expected: 'C', gradeC.getletterGrade( score: 70));
    }
    @Test
    public void letterGradePartitionD() {
        letterGrade gradeD = new letterGrade();
        assertEquals( expected: 'D', gradeD.getletterGrade( score: 69));
        assertEquals( expected: 'D', gradeD.getletterGrade( score: 65));
        assertEquals( expected: 'D', gradeD.getletterGrade( score: 60));
    }
    @Test
    public void letterGradePartitionF() {
        letterGrade gradeF = new letterGrade();
        assertEquals( expected: 'F', gradeF.getletterGrade( score: 59));
        assertEquals( expected: 'F', gradeF.getletterGrade( score: 30));
        assertEquals( expected: 'F', gradeF.getletterGrade( score: 0));
    }
    @Test
    public void letterGradePartitionX() {
        letterGrade gradeX = new letterGrade();
        assertEquals( expected: 'X', gradeX.getletterGrade( score: 101));
        assertEquals( expected: 'X', gradeX.getletterGrade( score: 110));
        assertEquals( expected: 'X', gradeX.getletterGrade( score: -1));
        assertEquals( expected: 'X', gradeX.getletterGrade( score: -10));
    }
}
```

Jacob › Git_test › Repository

master | git_test / SE / HW4 / src / main / **letterGradeTest.java** | Find file | Blame | History | Permalink

HW4 7(a)
Jacob authored 2 hours ago | 2617e686

letterGradeTest.java 1.75 KB | Edit | Web IDE | Replace | Delete

```java
package main;

import org.junit.jupiter.api.Test;

import static org.junit.Assert.assertEquals;

public class letterGradeTest {

    @Test
    public void letterGradePartitionA() {
        letterGrade gradeA = new letterGrade();
        assertEquals('A', gradeA.getletterGrade(100));
        assertEquals('A', gradeA.getletterGrade(95));
        assertEquals('A', gradeA.getletterGrade(90));
    }
    @Test
    public void letterGradePartitionB() {
        letterGrade gradeB = new letterGrade();
        assertEquals('B', gradeB.getletterGrade(89));
        assertEquals('B', gradeB.getletterGrade(85));
        assertEquals('B', gradeB.getletterGrade(80));
    }
```