1. Describe the 5 key activities in an object-oriented design process.
   (a) Define the context and modes of use of the system
   (b) Design the system architecture
   (c) Identify the principal system objects
   (d) Develop design models
   (e) Specify object interfaces

2. Consider software evolution?
   (a) 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. You use automated tools to process and reengineer a legacy system to create a new system that is more maintainable.

   Refactoring is 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) Describe the tree types of software maintenances?
      (1) Maintenance to repair software faults：
         甲、Changing a system to correct deficiencies in the way meets its requirements.
         乙、Corrective maintenance
      (2) Maintenance to adapt software to a different operating environment：
         甲、Changing a system so that it operates in a different environment (computer, OS, etc.) from its initial implementation.
         乙、Adaptive maintenance
      (3) Maintenance to add to or modify the system's functionality：
         甲、Modifying the system to satisfy new requirements.
         乙、Perfective maintenance

   (c) What are bad smells (or code smells)? How will you relate bad smells and preventative maintenance?
         甲、Duplicate code：The same or very similar code may be included at different places in a program. This can be removed and implemented as a single method or function that is called as required.

乙、Long methods：If a method is too long, it should be redesigned as a number of shorter methods.

丙、Switch (case) statements：These often involve duplication, where the switch depends on the type of a value. The switch statements may be scattered around a program. In object-oriented languages, you can often use polymorphism to achieve the same thing.

丁、Data clumping：Data clumps occur when the same group of data items (fields in classes, parameters in methods) re-occur in several places in a program. These can often be replaced with an object that encapsulates all of the data.

戊、Speculative generality：This occurs when developers include generality in a program in case it is required in the future. This can often simply be removed.

3. 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) Apply the equivalence partitioning testing technique to design test cases for testing the program.

Score range	grade	Test case
score >100	X	Score = 101
score>=90	A	Score = 95
80<=score<90	B	Score = 85
70<=score<80	C	Score = 75
60<=score<70	D	Score = 65
score<60	F	Score = 55
score<0	X	Score = -5

   (b) Based on your answers in (a), design additional test cases by applying the boundary value analysis testing technique.

Score range	Test case	期望 output
score >100	Score = 101	X
	Score = 100	A
	Score = 99	A
score >=90	Score = 91	A
	Score = 90	A
	Score = 89	B

80<=score<90	Score = 91	A
	Score = 90	A
	Score = 89	B
	Score = 81	B
	Score = 80	B
	Score = 79	C
70<=score<80	Score = 81	B
	Score = 80	B
	Score = 79	C
	Score = 71	C
	Score = 70	C
	Score = 69	D
60<=score<70	Score = 71	C
	Score = 70	C
	Score = 69	D
	Score = 61	D
	Score = 60	D
	Score = 59	F
score<60	Score = 61	D
	Score = 60	D
	Score = 59	F
score<0	Score = 1	F
	Score = 0	F
	Score = -1	X

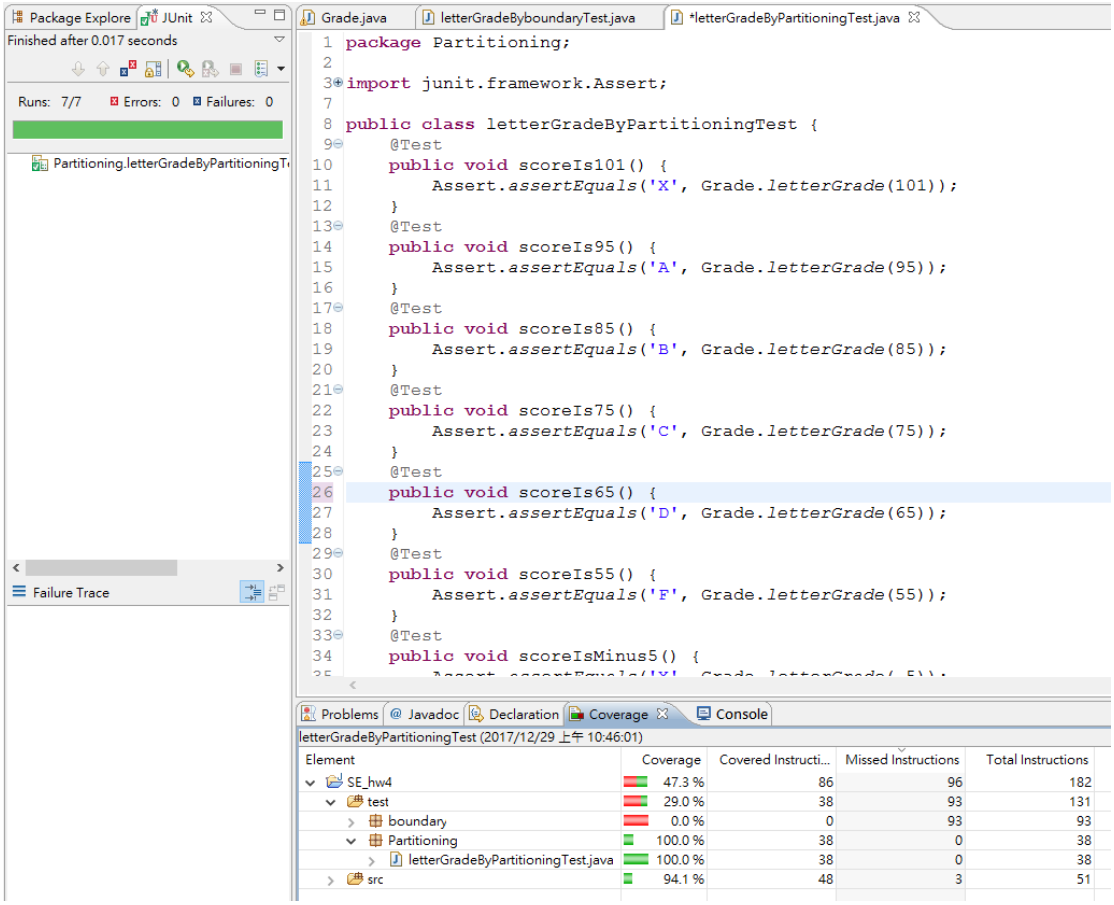
4. Consider the following program.

```
    public static char letterGrade(int score) {
1           char grade;
2           if (score <0 || score > 100)
3               grade = 'X';
4           else if (score>=90 && score <=100)
5               grade = 'A';
6           else if (score>=80 && score <90)
7               grade = 'B';
8           else if (score>=70 && score <80)
9               grade = 'C';
10          else if (score>=60 && score <70)
11              grade = 'D';
12          else
13              grade ='F';
14          return grade;
15      }
```

(a) 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).

```java
public static char letterGrade(int score) {
    char grade;
    if (score < 0 || score > 100)
        grade = 'X';
    else if (score >= 90 && score <= 100)
        grade = 'A';
    else if (score >= 80 && score < 90)
        grade = 'B';
    else if (score >= 70 && score < 80)
        grade = 'C';
    else if (score >= 60 && score < 70)
        grade = 'D';
    else
        grade = 'F';
    return grade;
}
```

```java
public class letterGradeByPartitioningTest {
    @Test
    public void scoreIs101() {
        Assert.assertEquals('X', Grade.letterGrade(101));
    }
    @Test
    public void scoreIs95() {
        Assert.assertEquals('A', Grade.letterGrade(95));
    }
    @Test
    public void scoreIs85() {
        Assert.assertEquals('B', Grade.letterGrade(85));
    }
    @Test
    public void scoreIs75() {
        Assert.assertEquals('C', Grade.letterGrade(75));
    }
    @Test
    public void scoreIs65() {
        Assert.assertEquals('D', Grade.letterGrade(65));
    }
    @Test
    public void scoreIs55() {
        Assert.assertEquals('F', Grade.letterGrade(55));
    }
    @Test
    public void scoreIsMinus5() {
        Assert.assertEquals('X', Grade.letterGrade(-5));
    }
}
```

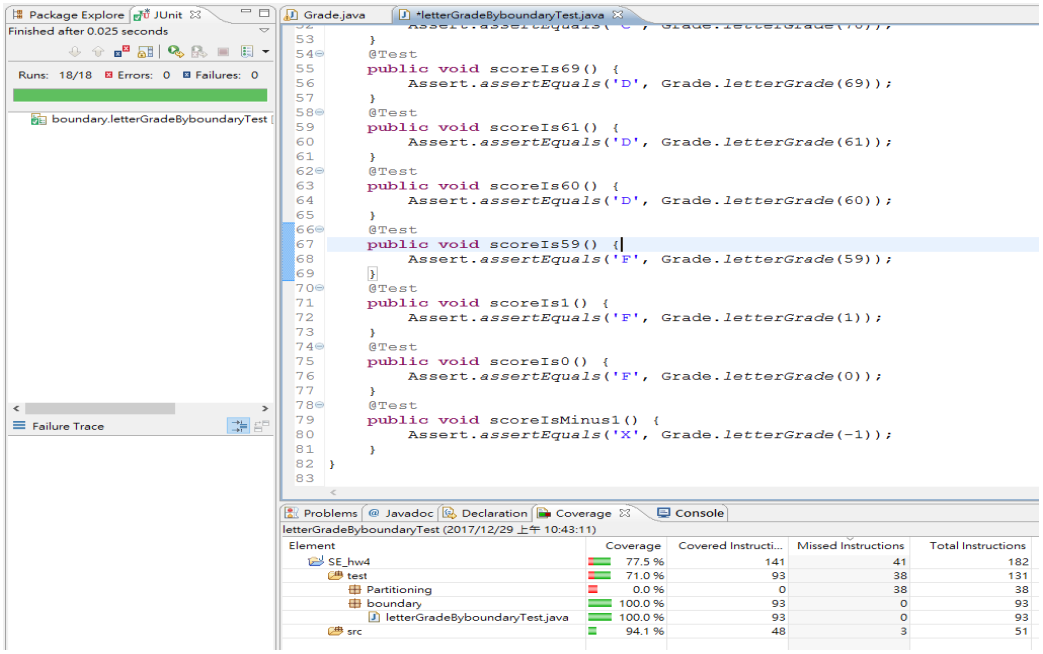


(b) Implement your test cases in problem 4(b) using JUnit. Show the JUnit source code of your test cases and the screen snapshots of the execution results of the test cases (including code coverage).

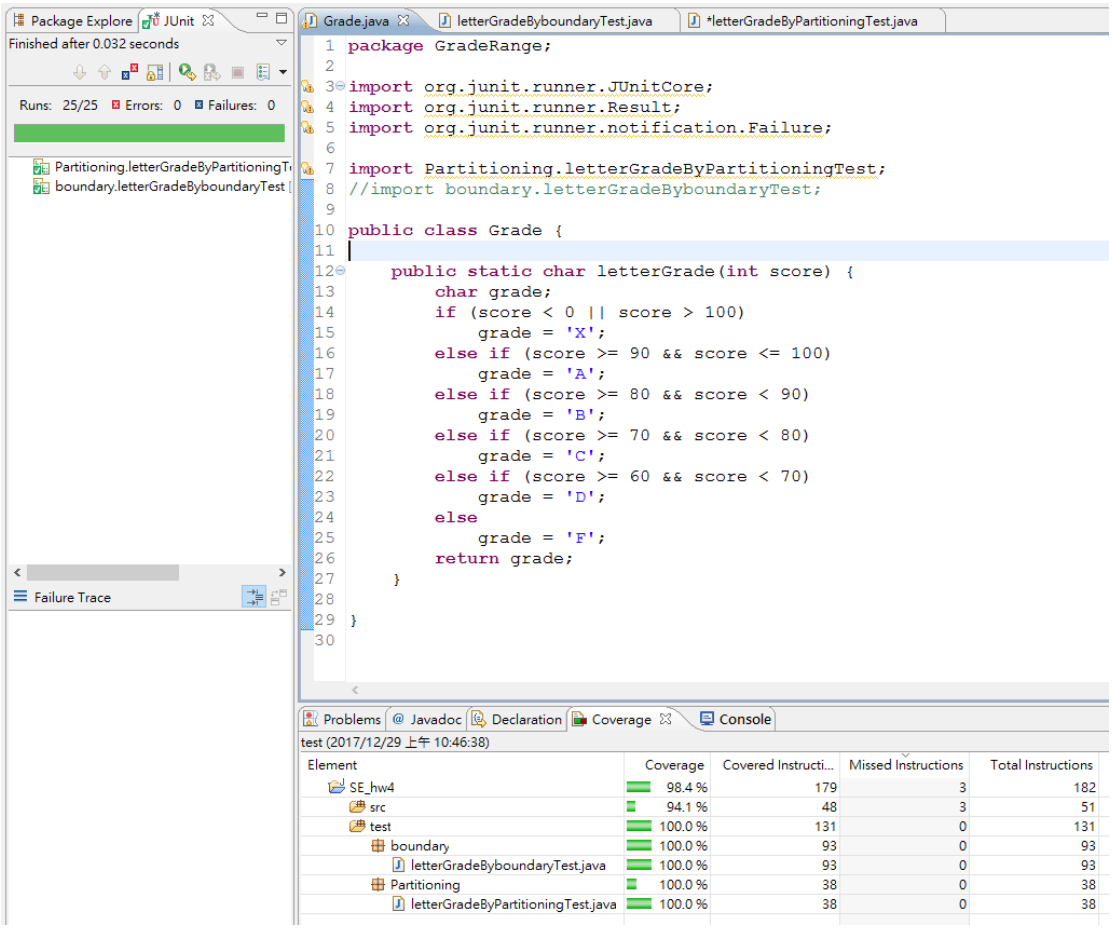
```
public class letterGradeByboundaryTest {
    @Test
    public void scoreIs101() {
        Assert.assertEquals('X', Grade.letterGrade(101));
    }
    @Test
    public void scoreIs100() {
        Assert.assertEquals('X', Grade.letterGrade(101));
    }
    @Test
    public void scoreIs99() {
        Assert.assertEquals('A', Grade.letterGrade(99));
    }
    @Test
    public void scoreIs91() {
        Assert.assertEquals('A', Grade.letterGrade(91));
    }
    @Test
    public void scoreIs90() {
        Assert.assertEquals('A', Grade.letterGrade(90));
    }
    @Test
    public void scoreIs89() {
        Assert.assertEquals('B', Grade.letterGrade(89));
    }
```

```
    @Test
    public void scoreIs81() {
        Assert.assertEquals('B', Grade.letterGrade(81));
    }
    @Test
    public void scoreIs80() {
        Assert.assertEquals('B', Grade.letterGrade(80));
    }
    @Test
    public void scoreIs79() {
        Assert.assertEquals('C', Grade.letterGrade(79));
    }
    @Test
    public void scoreIs71() {
        Assert.assertEquals('C', Grade.letterGrade(71));
    }
    @Test
    public void scoreIs70() {
        Assert.assertEquals('C', Grade.letterGrade(70));
    }
    @Test
    public void scoreIs69() {
        Assert.assertEquals('D', Grade.letterGrade(69));
    }
    @Test
    public void scoreIs61() {
        Assert.assertEquals('D', Grade.letterGrade(61));
    }
```

```
    @Test
    public void scoreIs60() {
        Assert.assertEquals('D', Grade.letterGrade(60));
    }
    @Test
    public void scoreIs59() {
        Assert.assertEquals('F', Grade.letterGrade(59));
    }
    @Test
    public void scoreIs1() {
        Assert.assertEquals('F', Grade.letterGrade(1));
    }
    @Test
    public void scoreIs0() {
        Assert.assertEquals('F', Grade.letterGrade(0));
    }
    @Test
    public void scoreIsMinus1() {
        Assert.assertEquals('X', Grade.letterGrade(-1));
    }
}
```



下圖為兩種測試方式都測的 code coverage



5. Illustrate the application of the JUnit and configuration management (CM) tools, such as subversion or Git (or GitHub), in software development. Noe 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) 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.

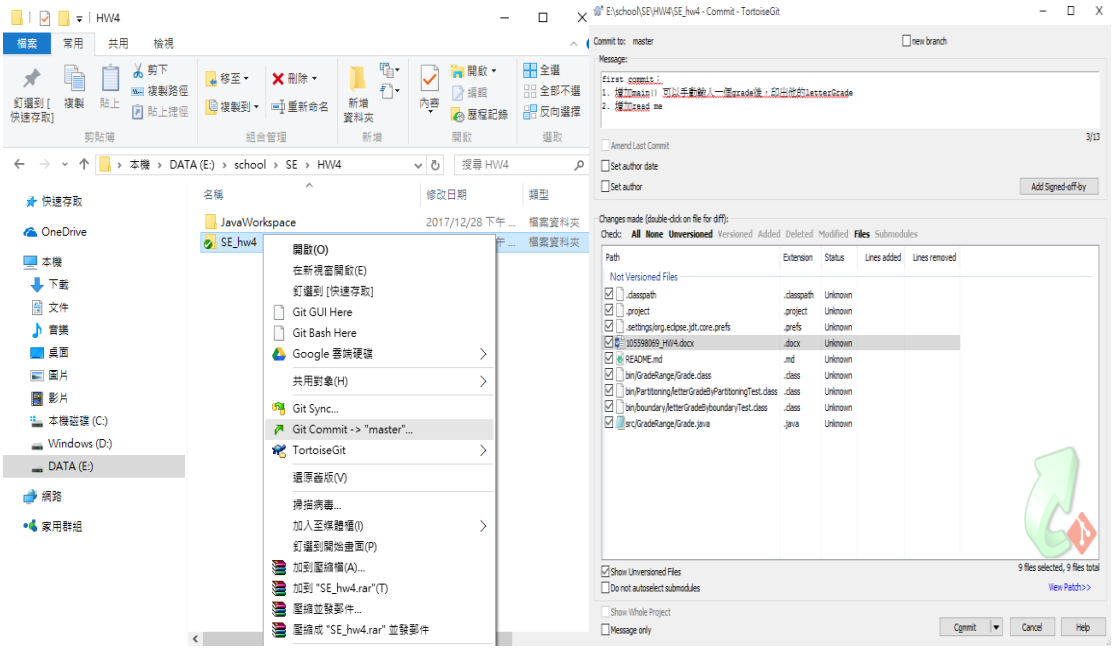
*Grade.java

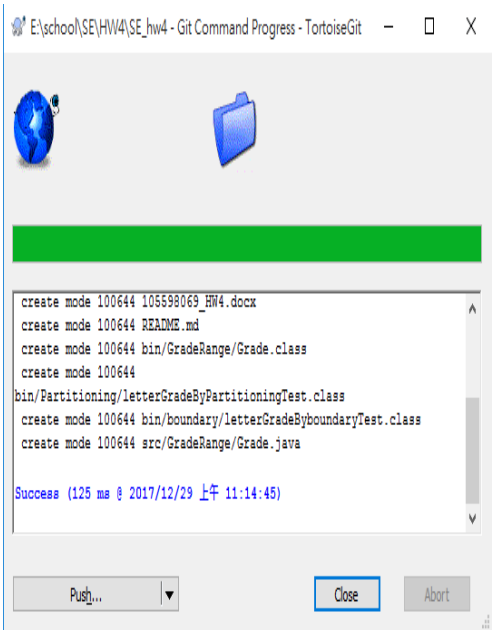
```java
package GradeRange;

import java.io.IOException;

public class Grade {
    public static void main(String[] args) throws IOException {
        Scanner sc = new Scanner(System.in);
        while(true) {
            System.out.print("Enter a grade:");
            String input=  sc.nextLine();
            if(input.equals("exit")){
                System.exit(0);
                sc.close();
            }
            int grade = Integer.parseInt(input);
            System.out.println("your letterGrade is : "+letterGrade(grade));
        }
    }

    public static char letterGrade(int score) {
        char grade;
        if (score < 0 || score > 100)
            grade = 'X';
        else if (score >= 90 && score <= 100)
            grade = 'A';
```

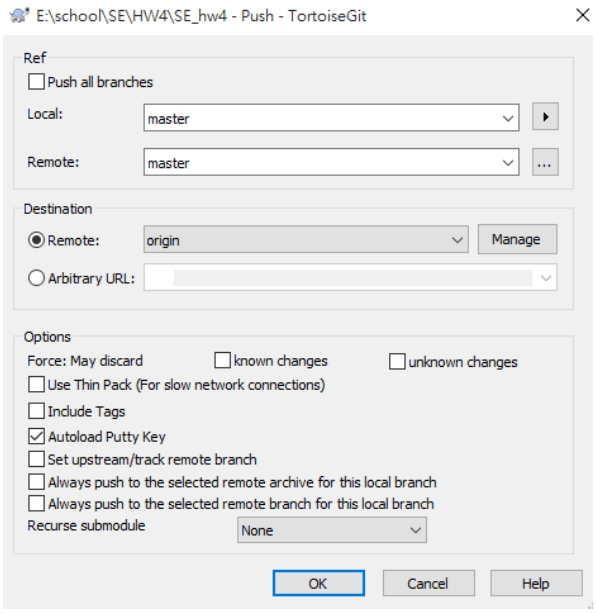
Problems | Javadoc | Declaration | Coverage | Console

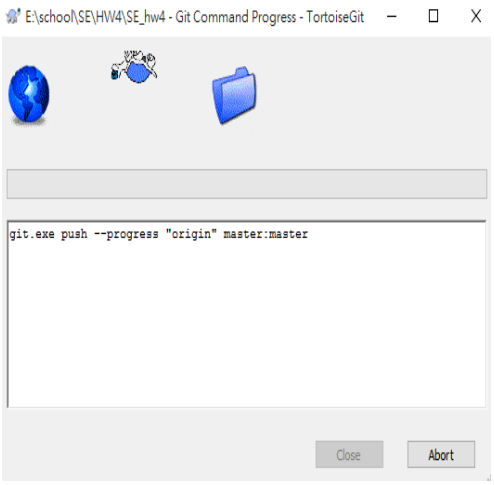
Grade (1) (2017/12/29 上午 11:26:21)

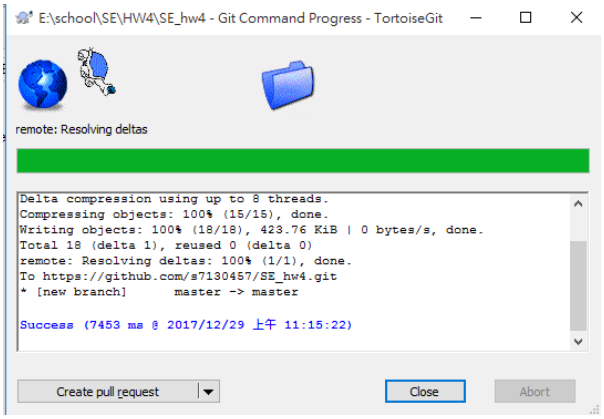
Element	Coverage	Covered Instructi...	Missed Instructions	Total Instructions
SE_hw4	35.8 %	77	138	215
test	0.0 %	0	131	131
src	91.7 %	77	7	84
GradeRange	91.7 %	77	7	84
Grade.java	91.7 %	77	7	84

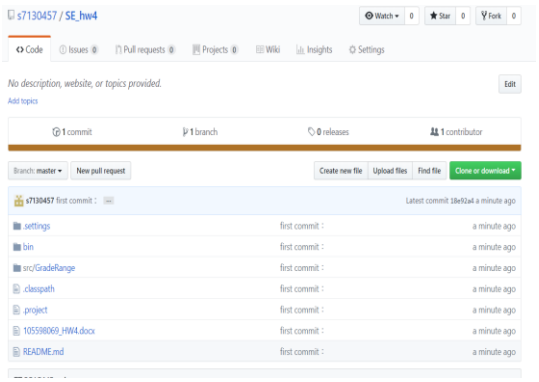




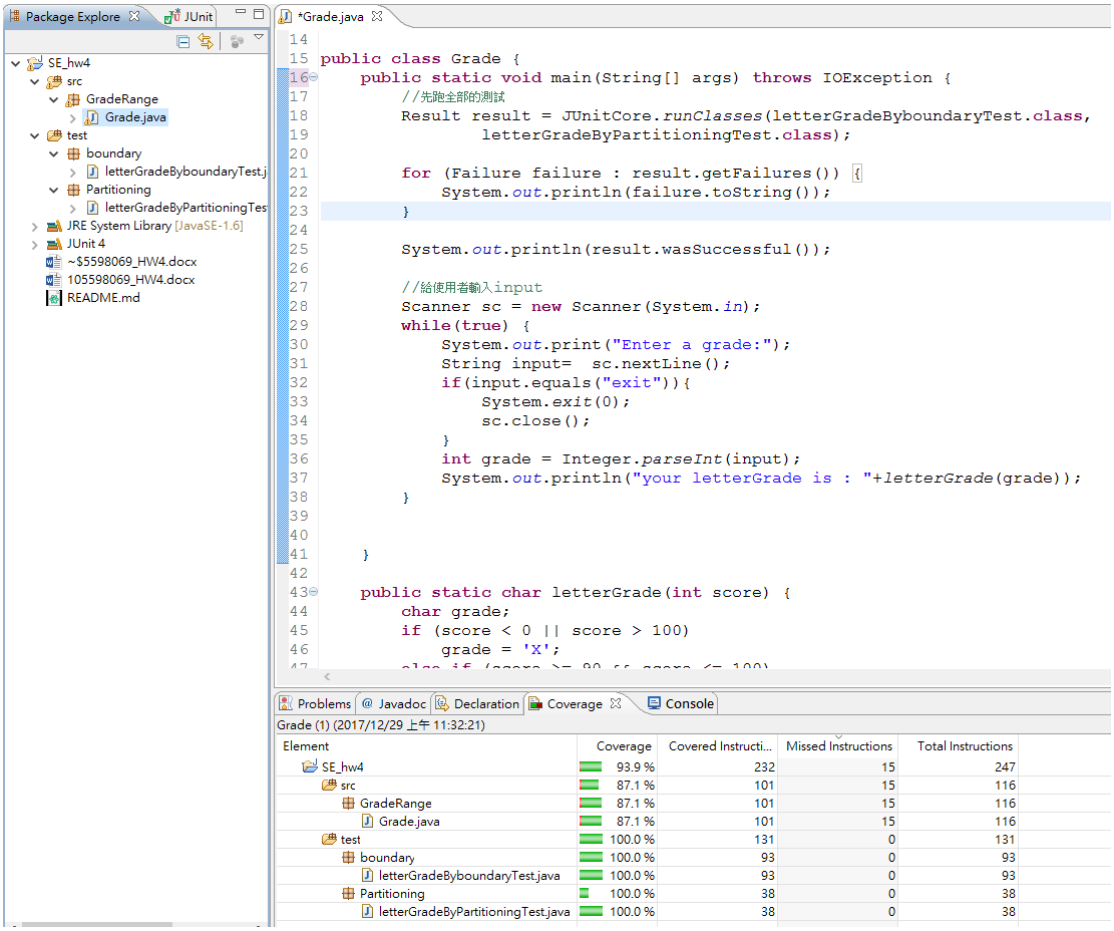


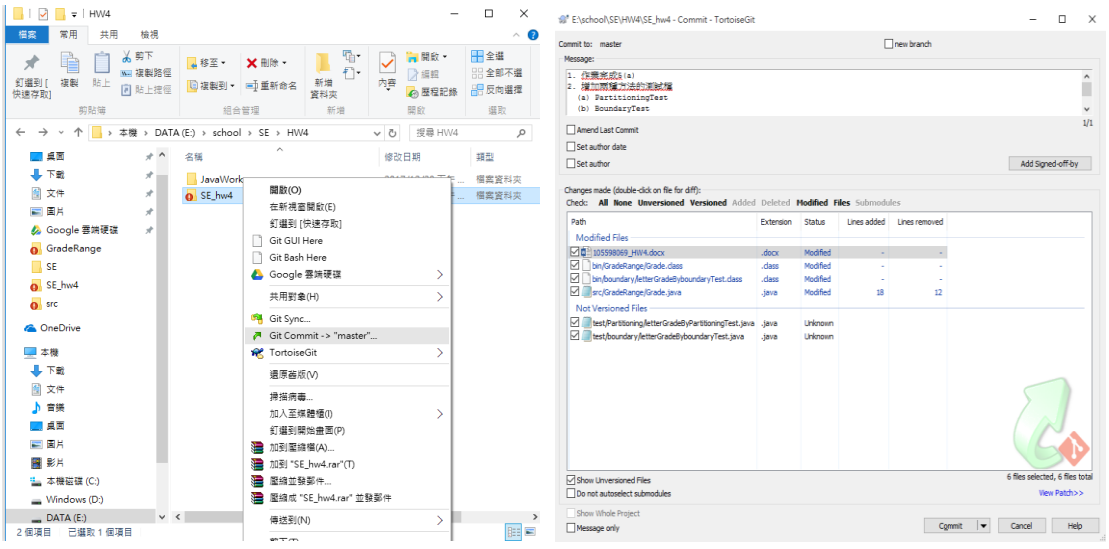


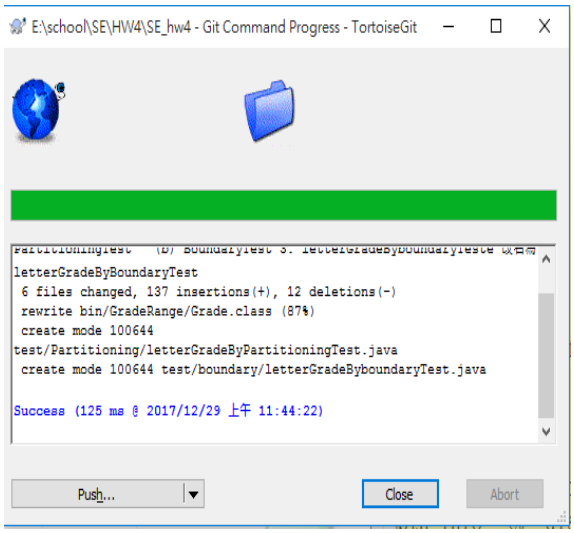


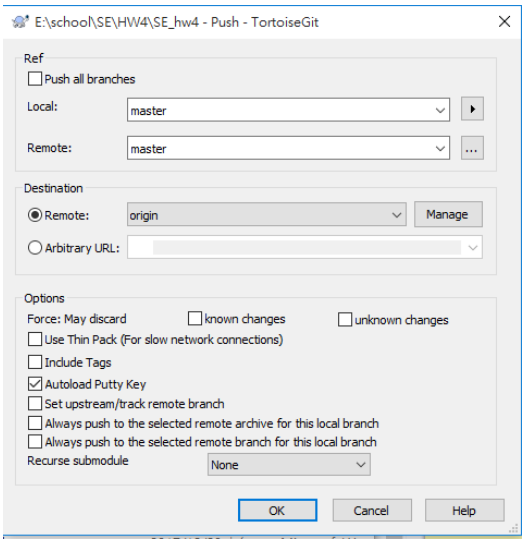


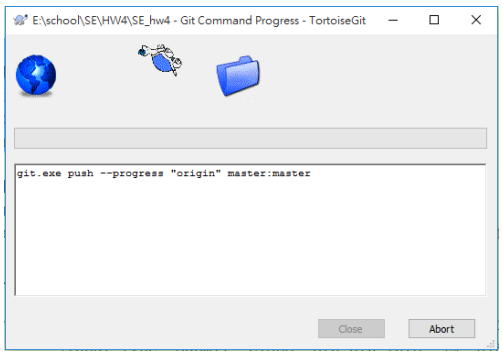
(b) Show the screen snapshots for using the CM and JUnit tools to check out your source code of letterGrade.java committed in 5(a) and add JUnit test cases to test the program automatically. After all the test cases are pass and the statement coverage is 100%, commit you source code and test cases to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

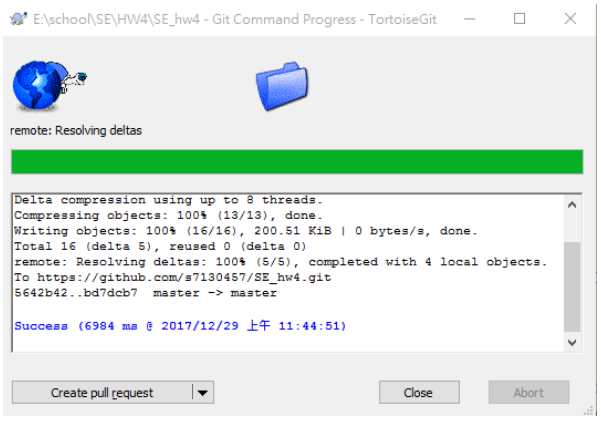

Package Explorer
JUnit
SE_hw4
src
GradeRange
Grade.java
test
boundary
letterGradeByboundaryTest.j
Partitioning
letterGradeByPartitioningTes
JRE System Library [JavaSE-1.6]
JUnit 4
~$5598069_HW4.docx
105598069_HW4.docx
README.md
*Grade.java
14
15 public class Grade {
16 public static void main(String[] args) throws IOException {
17 //先跑全部的測試
18 Result result = JUnitCore.runClasses(letterGradeByboundaryTest.class,
19 letterGradeByPartitioningTest.class);
20
21 for (Failure failure : result.getFailures()) {
22 System.out.println(failure.toString());
23 }
24
25 System.out.println(result.wasSuccessful());
26
27 //給使用者輸入input
28 Scanner sc = new Scanner(System.in);
29 while(true) {
30 System.out.print("Enter a grade:");
31 String input= sc.nextLine();
32 if(input.equals("exit")){
33 System.exit(0);
34 sc.close();
35 }
36 int grade = Integer.parseInt(input);
37 System.out.println("your letterGrade is : "+letterGrade(grade));
38 }
39
40
41 }
42
43 public static char letterGrade(int score) {
44 char grade;
45 if (score < 0 || score > 100)
46 grade = 'X';
Problems
Javadoc
Declaration
Coverage
Console
Grade (1) (2017/12/29 上午 11:32:21)
Element
Coverage
Covered Instructi...
Missed Instructions
Total Instructions
SE_hw4 93.9 % 232 15 247
src 87.1 % 101 15 116
GradeRange 87.1 % 101 15 116
Grade.java 87.1 % 101 15 116
test 100.0 % 131 0 131
boundary 100.0 % 93 0 93
letterGradeByboundaryTest.java 100.0 % 93 0 93
Partitioning 100.0 % 38 0 38
letterGradeByPartitioningTest.java 100.0 % 38 0 38


HW4
檔案
常用
共用
檢視
本機 › DATA (E:) › school › SE › HW4
搜尋 HW4
JavaWork
SE_hw4
開啟(O)
在新視窗開啟(E)
釘選到 [快速存取]
Git GUI Here
Git Bash Here
Google 雲端硬碟
共用對象(H)
Git Sync...
Git Commit -> "master"...
TortoiseGit
還原前版(V)
掃描病毒...
加入至媒體櫃(I)
釘選到開始畫面(P)
加到壓縮檔(A)...
加到 "SE_hw4.rar"(T)
壓縮並發郵件...
壓縮成 "SE_hw4.rar" 並發郵件
傳送到(N)
2 個項目 已選取 1 個項目
E:\school\SE\HW4\SE_hw4 - Commit - TortoiseGit
Commit to: master
new branch
Message:
Amend Last Commit
Set author date
Set author
Add Signed-off-by
Changes made (double-click on file for diff):
Check: All None Unversioned Versioned Added Deleted Modified Files Submodules
Path
Extension
Status
Lines added
Lines removed
Modified Files
105598069_HW4.docx .docx Modified
bin/GradeRange/Grade.class .class Modified
bin/boundary/letterGradeByboundaryTest.class .class Modified
src/GradeRange/Grade.java .java Modified 18 12
Not Versioned Files
test/Partitioning/letterGradeByPartitioningTest.java .java Unknown
test/boundary/letterGradeByboundaryTest.java .java Unknown
Show Unversioned Files
Do not autoselect submodules
6 files selected, 6 files total
View Patch>>
Show Whole Project
Message only
Commit
Cancel
Help









s7130457 / SE_hw4

Watch 0 | Star 0 | Fork 0

Code | Issues 0 | Pull requests 0 | Projects 0 | Wiki | Insights | Settings

No description, website, or topics provided.

Edit

Add topics

3 commits | 1 branch | 0 releases | 1 contributor

Branch: master | New pull request | Create new file | Upload files | Find file | Clone or download

s7130457 1. 作業完成5(a) ... Latest commit bd7dcb7 a minute ago

.settings	first commit：	30 minutes ago
bin	1. 作業完成5(a)	a minute ago
src/GradeRange	1. 作業完成5(a)	a minute ago
test	1. 作業完成5(a)	a minute ago
.classpath	first commit：	30 minutes ago
.project	first commit：	30 minutes ago
105598069_HW4.docx	1. 作業完成5(a)	a minute ago
README.md	first commit：	30 minutes ago