English Challenge

組員:資財三甲 陳順維
莊子弘

Problem Statement

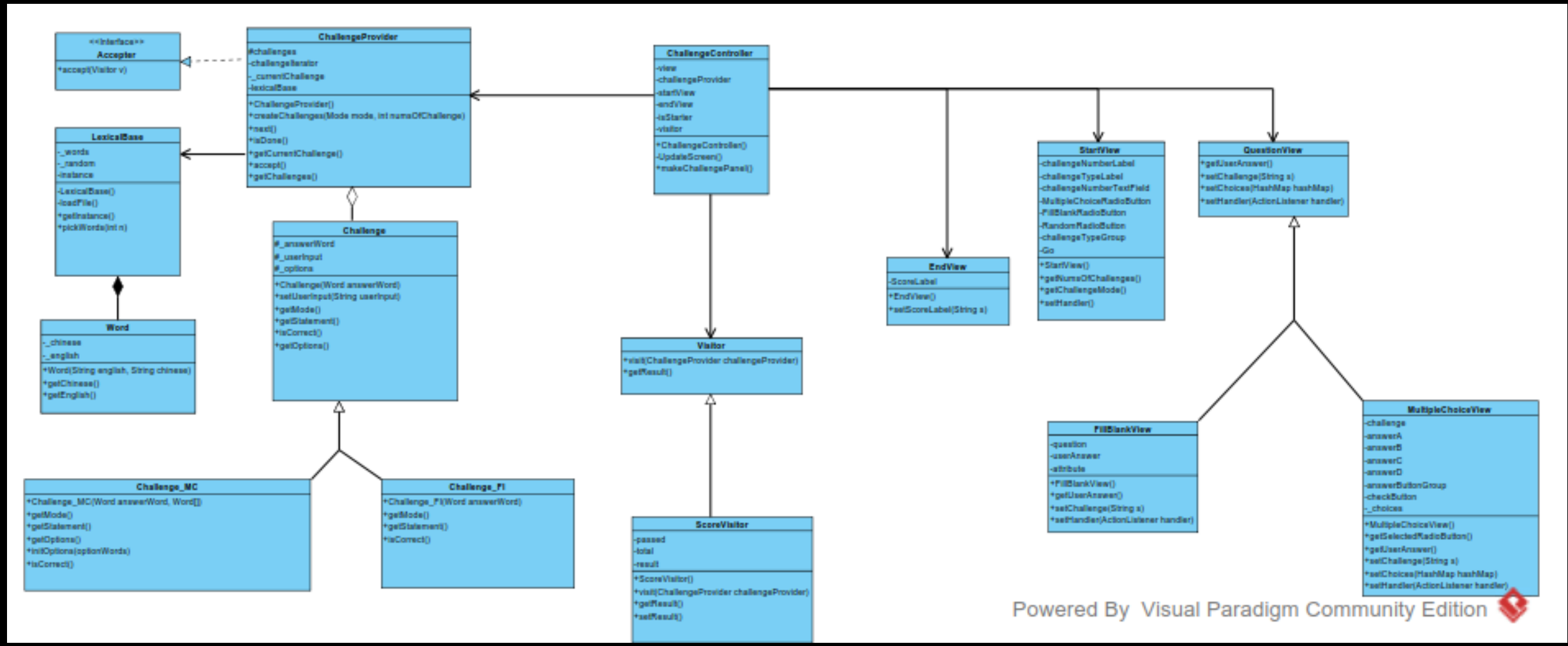
English Challenge是一個英文測驗的桌面應用程式，透過不同題型來測驗使用者，使用者可以:

- 選擇題數
- 選擇模式:全部選擇題、全部填空題、隨機挑選
- 顯示統計資料

Design Problems

- 若是將處理題目與處理UI的邏輯寫在一起，會使程式架構過於複雜，一個方法可能要同時處理UI與處理邏輯，難以維護。
- 當Client需要一個題目時，要自己從詞彙庫裡挑單字來創造題目，要根據不同題型創造不同題目，會使Client負擔額外工作，得知不必要的資訊。
- 如果要算答對題數的話，會讓負責處理題目邏輯的Class多負責其他操作。
- 如果有人重複產生了新的辭彙庫物件，將會消耗過多資源，需要避免重複產生新的辭彙庫。

Class Diagram



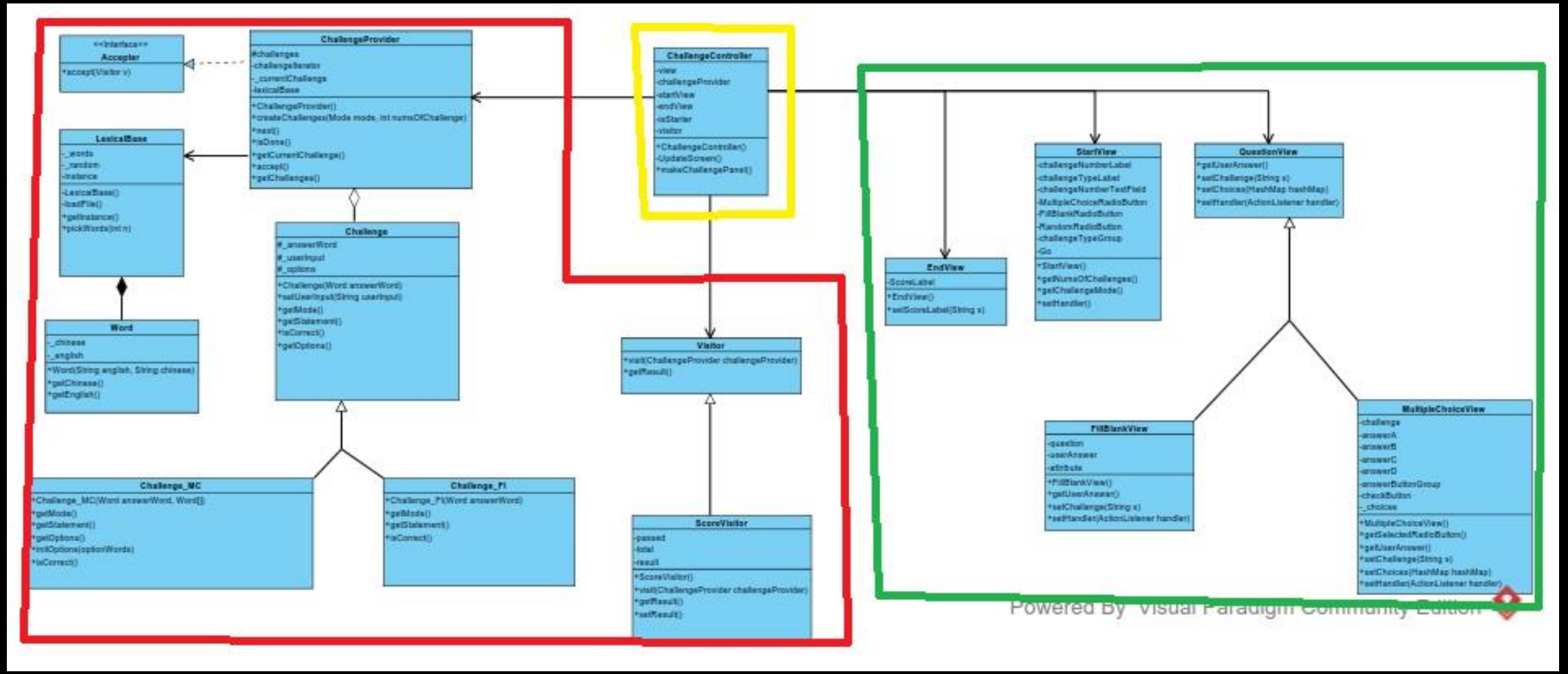
MVC - Why

- 若是將處理題目與處理UI的邏輯寫在一起，會使程式架構過於複雜，一個方法可能要同時處理UI與處理邏輯，難以維護。

MVC - Consequence

- 將處理UI的判斷與題目邏輯分開，使得程式碼容易維護
- 更容易分工合作
- 程式架構變大

MVC- Code Review



Simple Factory - Why

- 當Client需要一個題目時，要自己從詞彙庫裡挑單字來創造題目，要根據不同題型創造不同題目，會使Client負擔額外工作，得知不必要的資訊。

Simple Factory - Consequence

- Client不需要負擔額外工作，知道細節，就可以拿到題目。
- 增加了FactoryClass來負擔判別的責任

Factory - code review

```
public void createChallenges(Mode mode, int numsOfChallenge)
{
    for (int i = 0; i < numsOfChallenge; i++)
    {
        Challenge temp;
        Word word = lexicalBase.pickWords(1)[0];
        switch(mode)
        {
            case RAN:
                if ((int)(Math.random() * 2 + 1) == 1){
                    temp = new Challenge_FI(word);
                }
                else{
                    Word[] optionWords = lexicalBase.pickWords(3);
                    temp = new Challenge_MC(word, optionWords);
                }
                break;
            case MUL:
                Word[] optionWords = lexicalBase.pickWords(3);
                temp = new Challenge_MC(word, optionWords);
                break;
            case FILL:
                temp = new Challenge_FI(word);
                break;
            default:
                temp = null;
                break;
        }

        challenges.add(temp);
```

Visitor- Why

- 如果要算答對題數的話，會讓負責處理題目邏輯的Class多負責其他操作。
- 希望可以對主要功能的Class進行盡可能小的修改，但能實作新功能。

Visitor - Consequence

- 透過新增class達到新增額外的操作，不須讓原先的Class負擔

- 多了2個Class、1個Interface，設計變得複雜

Visitor - code review

```
4
5   public class ScoreVisitor extends Visitor {
6       int passed;
7       int total;
8       String result;
9
10  ›   public ScoreVisitor (){⋯
14
15      @Override
16      public void visit(ChallengeProvider challengeProvider) {
17
18          for (Challenge challenge: challengeProvider.getChallenges())
19          {
20              passed += challenge.isCorrect() ? 1 : 0;
21              total++;
22          }
23      }
24
25      @Override
26      public String getResult() {
27          setResult();
28          return result;
29      }
30
31  ›   private void setResult(){⋯
34  }
35
```

Singleton - Why

- 如果有人重複產生了新的辭彙庫物件，將會消耗過多資源，需要避免重複產生新的辭彙庫。

Singleton - Consequence

- 確保耗資源且只需單一物件的Class不會被創造出多餘的物件，避免消耗無謂資源
- 如果當這個物件真的需要多個的時候，會導致程式有BUG

Singleton - code review

```
10  public class LexicalBase {
11      private static LexicalBase instance = new LexicalBase();
12
13      private ArrayList<Word> _words;
14      private Random _random;
15
16
17  >   private LexicalBase(){…
29
30  >   private void loadFile()throws IOException {…
44
45  >   public Word[] pickWords(int n){…
56
57      public static LexicalBase getInstance(){
58          return instance;
59      }
60
61  }
62
```

problem resolved

- did I solve the problem on page 2?
  YES，已經實作出Page 2的所有功能

- did I solve the design problems on page 3?
  YES，已經將Page 3的設計問題解決了，雖然不可避免地帶有一些副作用