**影音管理軟體**

**HW# 5**

四電資四 101820302 施帛辰

四電資四 101820340 鄒令業

1. **Requirement Document**
   1. **Change History**

|  |  |  |
| --- | --- | --- |
| **Iteration I** | | |
| **Version** | **Description** | **Date** |
| 1 | Cover Page  Problem Statement  The Development Language | 2016.2.28 |
| 2.1 | Change History  System Feature  Use Case 1 | 2016.3.10 |
| 2.2 | Modify Use Case 1  Add Use Case 2 & 3 | 2016.3.15 |
| 2.3 | Add Use Case 4 | 2016.3.17 |
| 3.1 | Domain Diagram Design | 2016.3.29 |
| 4.1 | Logic Architecture & SSD | 2016.4.12 |
| 4.2 | Contract & Operation Sequence Diagram  SSD modified | 2016.4.26 |
| 4.3 | Class Diagram | 2016.4.27 |
| 5.1 | Add Initialize Seq and Destructor Seq and Implementation Class Diagram | 2016.5.03 |
| 5.2 | Add the difference between implementation and design and the information of source code | 2016.5.05 |

* 1. **Problem Statement**

日前電視劇風靡各年齡層，每周推出的新集數與每季上映的新劇，往往令那些電視劇觀賞者眼花撩亂，也讓那些想嘗試新劇的人不知如何下手，於是我們想到利用一套軟體記錄使用者的觀賞進度與喜好程度，同時管理使用者的影集，並可利用軟體推薦使用者可能喜歡的新劇，而使用者可在自家使用任意設備、平台進行記錄。

因為電視劇眾多且集數與觀賞記錄個不同，因此使用者容易遺失自己的影集進度，也不容易清楚究竟哪些新劇符合自己的胃口，透過此軟體，在每一次觀看前便可查詢先前的進度，而觀看後做上記錄與評論，就可以避免進度的遺失，也可藉評論讓軟體歸納使用者可能喜歡的類型，進而推薦新劇給有需要的人。

本軟體提供GUI，讓使用者快速、簡便、直覺地操作，並記錄影集的觀看進度，也讓使用者可清楚自己喜好的類別與觀看歷史。

* 1. **System Context Diagram**



* 1. **System Features**

1. 管理影集資訊
2. 追蹤影集
3. 撰寫觀後心得
4. 推薦影集
   1. **Use Case Diagram**



* 1. **Use Case**

1. Use Case 1

|  |  |
| --- | --- |
| Use Case Name | 管理影集資訊 |
| Scope | 影音管理軟體 |
| Level | User Goal |
| Prime Actor | 觀看者 |
| Stakeholder and Interests | 觀看者：管理影集的資訊，包含取得、新增、修改、刪除影集資訊 |
| Preconditions | 觀看者已安裝軟體 |
| Success Guarantee | 觀看者能成功管理影集資訊，並看見結果 |
| Main Success Scenario | 1. 觀看者啟動軟體 2. 軟體自行抓取最新資訊 3. 觀看者手動對影集資訊進行管理 4. 顯示正確結果 |
| Extensions | \*a. 在任何情況下，若軟體無預警關閉  1. 觀看者重新啟動軟體。  2. 軟體將未處理完的資訊清除  3. 回到第二步  2a. 沒有網路的情況下，通知觀看者，目前裝置尚未連接網路  2b. 如果沒有新的影集資訊，通知觀看者  3a. 若觀看者要新增影集資訊   1. 觀看者使用新增功能 2. 觀看者輸入影集資訊 3. 觀看者完成新增   3b. 若觀看者要匯入影集資訊   1. 觀看者使用匯入功能 2. 觀看者選擇要匯入的檔案 3. 軟體完成匯入   3c. 若觀看者要修改影集資訊   1. 觀看者選擇影集並修改 2. 觀看者輸入修改的資訊 3. 觀看者完成修改   3d. 若觀看者要刪除影集資訊   1. 觀看者選擇影集並刪除 2. 軟體再次確認影集的刪除   2a. 若觀看者確認刪除，軟體刪除影集資訊 |
| Special Requirements | NFR-01、NFR-02、NFR-03 |
| Technology and Data Variations List | 網路影集資訊與私人影集資訊擁有個別的獨立編號。  影集資訊中，包含描述與類別。 |
| Frequency of Occurrence | 每次啟動後一定會發生至少一次 |
| Open Issue | 1. 影集資料格式尚未決定 2. 各部影集的獨立編號產生方式尚未決定 3. 伺服器是要租用還要自己架設 4. 匯入的資料格式尚未決定 |

1. Use Case 2

|  |  |
| --- | --- |
| Use Case Name | 追蹤影集 |
| Scope | 影音管理軟體 |
| Level | User Goal |
| Prime Actor | 觀看者 |
| Stakeholder and Interests | 觀看者：對影集進行追縱，包含新增追蹤的影集、修改追蹤進度、取消追蹤 |
| Preconditions | 觀看者已安裝軟體 |
| Success Guarantee | 觀看者能成功追蹤影集，並看見結果 |
| Main Success Scenario | 1. 觀看者選擇影集 2. 軟體顯示影集資訊 3. 觀看者使用追蹤功能 4. 顯示正確結果 |
| Extensions | 3a. 若觀看者要追蹤新的影集，軟體紀錄開始追蹤  3b. 若觀看者要新增已追蹤的影集集數   1. 觀看者使用新增集數功能 2. 觀看者輸入集數資訊 3. 軟體顯示該影集的集數資訊   重複2、3步驟，直到觀看者不再新增集數  3c. 若觀看者要修改已追蹤的影集進度   1. 觀看者使用修改進度功能 2. 觀看者紀錄觀看的集數   2a. 若集數不存在，觀看者新增集數   1. 軟體要求輸入評論   3a. 若觀看者取消輸入，則不新增評論  3b. 若觀看者輸入評論，則新增一筆評論  3d. 若觀看者要取消已追蹤的影集   1. 觀看者使用取消追蹤的功能 2. 軟體再去確認影集取消追蹤   2a. 若觀看這確認，軟體取消影集的追蹤  3e. 若觀看者要恢復已取消追蹤的影集   1. 軟體恢復開始追蹤 2. 軟體讀取先前的集數資訊，並顯示 |
| Special Requirements | NFR-03、NFR-01 |
| Technology and Data Variations List | NA |
| Frequency of Occurrence | 經常發生 |
| Open Issue | NA |

1. Use Case 3

|  |  |
| --- | --- |
| Use Case Name | 撰寫觀後心得 |
| Scope | 影音管理軟體 |
| Level | User Goal |
| Prime Actor | 觀看者 |
| Stakeholder and Interests | 觀看者：可以記錄自己的觀後心得 |
| Preconditions | 觀看者至少有一部已追蹤的影集 |
| Success Guarantee | 觀看者能成功紀錄下觀後心得 |
| Main Success Scenario | 1. 觀看者選擇影集 2. 軟體顯示影集資訊 3. 觀看者開始撰寫心得 4. 軟體定期儲存當前的心得資訊 5. 觀看者結束心得的撰寫 |
| Extensions | 3a. 若軟體發現上次沒有正確儲存的心得資料   1. 軟體詢問觀看者是否重新載入上次的心得   1a. 若觀看者確認，則軟體顯示上次心得  1b. 若觀看者取消，則軟體清除上次心得  5a. 若觀看者取消心得撰寫   1. 軟體詢問觀看者是否保留目前的心得   1a. 若觀看者確認，軟體保留當前心得記錄  1b. 若觀看者取消，軟體清除定期儲存的心得資訊  5b. 若觀看者完成心得   1. 軟體詢問觀看者是否儲存心得   1a. 若觀看者確認，軟體儲存並完成心得  1b. 若觀看者取消，觀看者可以繼續編輯心得 |
| Special Requirements | NFR-03、NFR-01 |
| Technology and Data Variations List | NA |
| Frequency of Occurrence | 經常發生 |
| Open Issue | NA |

1. Use Case 4

|  |  |
| --- | --- |
| Use Case Name | 推薦影集 |
| Scope | 影音管理軟體 |
| Level | User Goal |
| Prime Actor | 軟體 |
| Stakeholder and Interests | 觀看者：希望可以看見軟體所推薦的影集  軟體：推薦影集給觀看者 |
| Preconditions | 觀看者可以有過去影集的追蹤紀錄 |
| Success Guarantee | 影集被推薦給觀看者 |
| Main Success Scenario | 1. 觀看者使用推薦功能 2. 軟體顯示數個推薦影集 3. 觀看者對推薦影集操作 |
| Extensions | 2a, 若軟體無法取得觀看者的資料   1. 軟體通知觀看者，無法推薦影集   2b. 若軟體的推薦影集皆被觀看者列入黑名單   1. 軟體通知觀看者，無非黑名單的推薦影集   3a. 若觀看者對某推薦影集有興趣   1. 觀看者對該部影集進行追蹤   3b. 若觀看者對某推薦影集不感興趣   1. 觀看者取消推薦該影集 2. 軟體將該影集列入黑名單   3c. 若觀看者希望再次推薦   1. 觀看者使用再次推薦功能 2. 回到步驟二   3d. 若觀看者不進行任何操作，直接離開 |
| Special Requirements | NFR-01、NFR-03 |
| Technology and Data Variations List | 依照影集類別，軟體一次最多推薦5部影集 |
| Frequency of Occurrence | 偶爾發生 |
| Open Issue | NA |

* 1. **Non-Functional Requirement and Constraints**

|  |  |  |
| --- | --- | --- |
| NFR ID | Category | Description |
| NFR-01 | Performance | 資料讀寫需要在一秒內完成 |
| NFR-02 | Performance | 伺服器要在0.5秒內回應 |
| NFR-03 | Usability | 通知要夠大夠清楚 |
| NFR-04 | Usability | UI 要足夠友善 |
| NFR-05 | Reliability | 資料讀寫必須正確無誤 |

* 1. **Glossary**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Term | Definition and Information | Format | Validation Rules | Aliases |
| 影集 | 以單集為播放單位而長期放映的影片 |  |  | 影劇、Series |
| 集數 | 一部影集的最小單位 |  |  | Episode |

* 1. **Software Environments**

The program will be written in C# language with Visual Studio.

1. **Domain Class Model**
   1. **Domain Class Diagram Showing Only Concepts**
      1. **Class Identified**

* Business Transaction : Series、Episode、Blacklist、Tracing\_List、Abandoned\_List
* Products : Review、Command
* Description : Series\_Description、Episode\_Description
* Catalogs : Catalog
* Collaborating System : Server、Software

P.S. 以上為使用類別清單 (Catalog List) 所找出來的Concepts，其中有些Concepts並沒有劃入Domain Model中。

* + 1. **Bad Class**

以Attribute的方式取代：

Series\_Description, Episode\_Description, BlackList, Tracing\_List, Abandoned\_List

* + 1. **Good Class**

Series, Episode：Domain基礎物件

Review, Command：功能之一

Category：分類影音的物件

Server：負責外部連結的物件

Software：Root Object

* 1. **Add Associations**
  2. **Add Attributes**

1. **Design**
   1. **Logic Architecture**
   2. **Use-Case Realizations with GRASP Patterns**
      1. **System Sequence Diagram**



* + 1. **Contract**

|  |  |
| --- | --- |
| Contract ID | Operation Name |
| CO-01 | AddSeries |
| CO-02 | ImportFile |
| CO-03 | SelectSeries |
| CO-04 | ModifySeries |
| CO-05 | RemoveSeries |

* + - 1. **AddSeries**

|  |  |
| --- | --- |
| Operation | AddSeries(name:string, description:string) |
| Cross Reference | Use Case1 |
| Preconditions | Software was opened |
| Postconditions | A new series instance s was created.(instance creation)  S was added into series list of software.(association formed) |

* + - 1. **ImportFile**

|  |  |
| --- | --- |
| Operation | ImportFile(filePath:string) |
| Cross Reference | Use Case1 |
| Preconditions | Software was opened |
| Postconditions | A list of new series sl was created.(instance creation)  Sl was added into series list of software.(association formed) |

* + - 1. **SelectSeries**

|  |  |
| --- | --- |
| Operation | SelectSeries(sid: int) |
| Cross Reference | Use Case1 |
| Preconditions | Software was opened |
| Postconditions | Software.selectedSeries became a series s.(attribute modification) |

* + - 1. **ModifySeries**

|  |  |
| --- | --- |
| Operation | ModifySeries (name: string, description: string) |
| Cross Reference | Use Case1 |
| Preconditions | A series s has been selected. |
| Postconditions | s.name was modified.(instance creation)  s.description was modified.(attribute modification) |

* + - 1. **RemoveSeries**

|  |  |
| --- | --- |
| Operation | RemoveSeries (sid: int) |
| Cross Reference | Use Case1 |
| Preconditions | Software was opened |
| Postconditions | A series s was removed from the list of series of software.(attribute modification) |

* + 1. **Operation Sequence Diagram**
       1. **AddSeries**
       2. **ImportFile**



* + - 1. **SelectSeries**



* + - 1. **ModifySeries**



* + - 1. **RemoveSeries**
      2. **Initialize**
      3. **Destroy**
  1. **Design Class Diagram**

1. **Implementation Class Model**
   1. **Implementation Class Diagram**
   2. **The Difference between Implementation and Design Class Model**
      1. **Comparison with Design and Implementation Class**

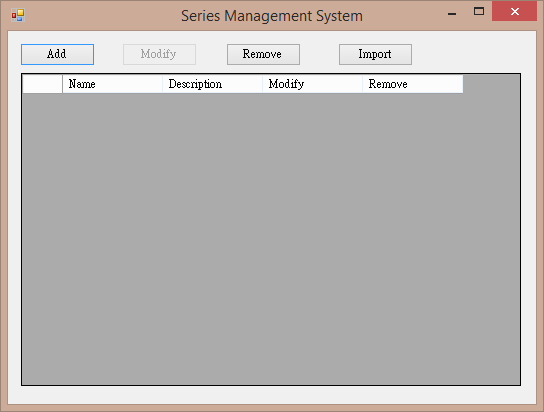
|  |  |  |  |
| --- | --- | --- | --- |
| **Class** | **Method** | **Design** | **Imp.** |
| **Software** | **AddSeries** | **Yes** | **Yes** |
| **ImportFile** | **Yes** | **Yes** |
| **SelectSeries** | **Yes** | **Yes** |
| **ModifySeries** | **Yes** | **Yes** |
| **RemoveSeries** | **Yes** | **Yes** |
| **FileManager** | **ImportFile** | **Yes** | **Yes** |
| **GetList** | **Yes** | **Fixed** |
| **SaveFile** | **No** | **New** |
| **Series** | **SetName** | **Yes** | **Yes** |
| **SetDescription** | **Yes** | **Yes** |
| **SetSeriesID** | **No** | **New** |
| **SeriesManager**  **(add in Imp. phase)** | **GetSeriesList** | **No** | **New** |
| **GetSelectedSeries** | **No** | **New** |
| **AddSeries** | **No** | **New** |
| **AddList** | **No** | **New** |
| **SelectSeries** | **No** | **New** |
| **ModifySelectedSeries** | **No** | **New** |
| **RemoveSeries** | **No** | **New** |
| **InitializeCount** | **No** | **New** |
| **Server**  **(add in Imp. phase)** | **GetData** | **No** | **New** |

* + 1. **Summary of Implementation Class / Method Changed**

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Number of Added** | **Number of Removed** | **Number of Modified** |
| **Class** | **2** | **0** | **1** |
| **Method** | **11** | **0** | **1** |

* 1. **The Lines of Code**

|  |  |  |  |
| --- | --- | --- | --- |
| **No** | **Class Name** | **Number of Methods** | **Line of codes without comment** |
| **1** | **Software** | **5** | **14** |
| **2** | **FileManger** | **3** | **20** |
| **3** | **Series** | **3** | **14** |
| **4** | **SeriesManager** | **8** | **27** |
| **5** | **Server** | **1** | **12** |

1. **Programing**
   1. **Snapshot of System Execution**
   2. **Source Code Listing**
      1. **Series**

public class Series

{

private int \_seriesID;

private string \_name;

private string \_description;

public Series(string name, string description)

{

\_name = name;

\_description = description;

}

[JsonConstructor]

public Series(string name, string description, int seriesID) :

this(name, description)

{

\_seriesID = seriesID;

}

#region Public Properties

public string Name

{

get

{

return \_name;

}

set

{

\_name = value;

}

}

public string Description

{

get

{

return \_description;

}

set

{

\_description = value;

}

}

public int SeriesID

{

get

{

return \_seriesID;

}

set

{

\_seriesID = value;

}

}

#endregion

}

* + 1. **SeriesManager**

public class SeriesManager

{

private List<Series> \_series = null;

private Series \_selectedSeries;

private int \_count = 0;

#region Constructor

public SeriesManager(List<Series> list)

{

\_series = list;

InitializeCount();

}

#endregion

#region Public Object

public List<Series> SeriesList

{

get

{

return \_series;

}

}

public Series SelectedSeries

{

get

{

return \_selectedSeries;

}

}

#endregion

public void AddSeries(String name, String description)

{

Series series = new Series(name, description, \_count++);

\_series.Add(series);

}

public void AddList(List<Series> list)

{

foreach (Series series in list)

{

series.SeriesID = \_count++;

}

\_series.AddRange(list);

}

public void SelectSeries(int sid)

{

\_selectedSeries = \_series.Find((x) => x.SeriesID == sid);

}

public void ModifySelectedSeries(string newName, string newDescription)

{

\_selectedSeries.Name = newName;

\_selectedSeries.Description = newDescription;

}

public void RemoveSeries(int sid)

{

Series series = \_series.Find((s) => s.SeriesID == sid);

\_series.Remove(series);

}

private void InitializeCount()

{

if (\_series.Count != 0)

{

\_count = \_series[\_series.Count - 1].SeriesID + 1;

}

}

}

* + 1. **Software**

public class Software

{

private SeriesManager \_seriesManager;

private FileManager \_fileManager;

public Software()

{

\_fileManager = new FileManager();

List<Series> list = \_fileManager.GetList();

\_seriesManager = new SeriesManager(list);

}

// Add a new series with name and description.

public void AddSeries(string name, string description)

{

\_seriesManager.AddSeries(name, description);

}

//Import series data from a file.

public void ImportFile(string filePath)

{

\_fileManager.ImportFile(filePath);

List<Series> list = \_fileManager.GetList();

\_seriesManager.AddList(list);

}

public void SelectSeries(int sid)

{

\_seriesManager.SelectSeries(sid);

}

public void ModifySeries(string newName, string newDescription)

{

\_seriesManager.ModifySelectedSeries(newName, newDescription);

}

public void RemoveSeries(int sid)

{

\_seriesManager.RemoveSeries(sid);

}

~Software()

{

List<Series> list = \_seriesManager.SeriesList;

\_fileManager.SaveFile(list);

}

}

* + 1. **FileManager**

public class FileManager

{

private List<Series> \_list;

private const string LOCAL\_STOREAGE = "./dat/data.dat";

public FileManager()

{

try

{

ImportFile(LOCAL\_STOREAGE);

}

catch (Exception e)

{

if (e is FileNotFoundException || e is DirectoryNotFoundException)

{

\_list = new List<Series>();

}

else

throw e;

}

}

public void ImportFile(string filePath)

{

String fileContext;

using (var streamReader = new StreamReader(filePath, Encoding.UTF8))

{

fileContext = streamReader.ReadToEnd();

}

\_list = JsonConvert.DeserializeObject<List<Series>>(fileContext) as List<Series>;

}

public List<Series> GetList()

{

return \_list;

}

public void SaveFile(List<Series> series)

{

new FileInfo(LOCAL\_STOREAGE).Directory.Create();

using (FileStream fs = File.OpenWrite(LOCAL\_STOREAGE))

{

string data = JsonConvert.SerializeObject(series);

Byte[] info = new UTF8Encoding(true).GetBytes(data);

fs.Write(info, 0, info.Length);

fs.Close();

}

}

}

* + 1. **Server**

public class Server

{

public string GetData()

{

string data;

HttpWebRequest request = (HttpWebRequest)HttpWebRequest.Create(SERVER\_URL);

request.Method = "GET";

using (WebResponse wr = request.GetResponse())

{

using (Stream s = wr.GetResponseStream())

{

using (StreamReader sr = new StreamReader(s, Encoding.UTF8))

{

data = sr.ReadToEnd();

}

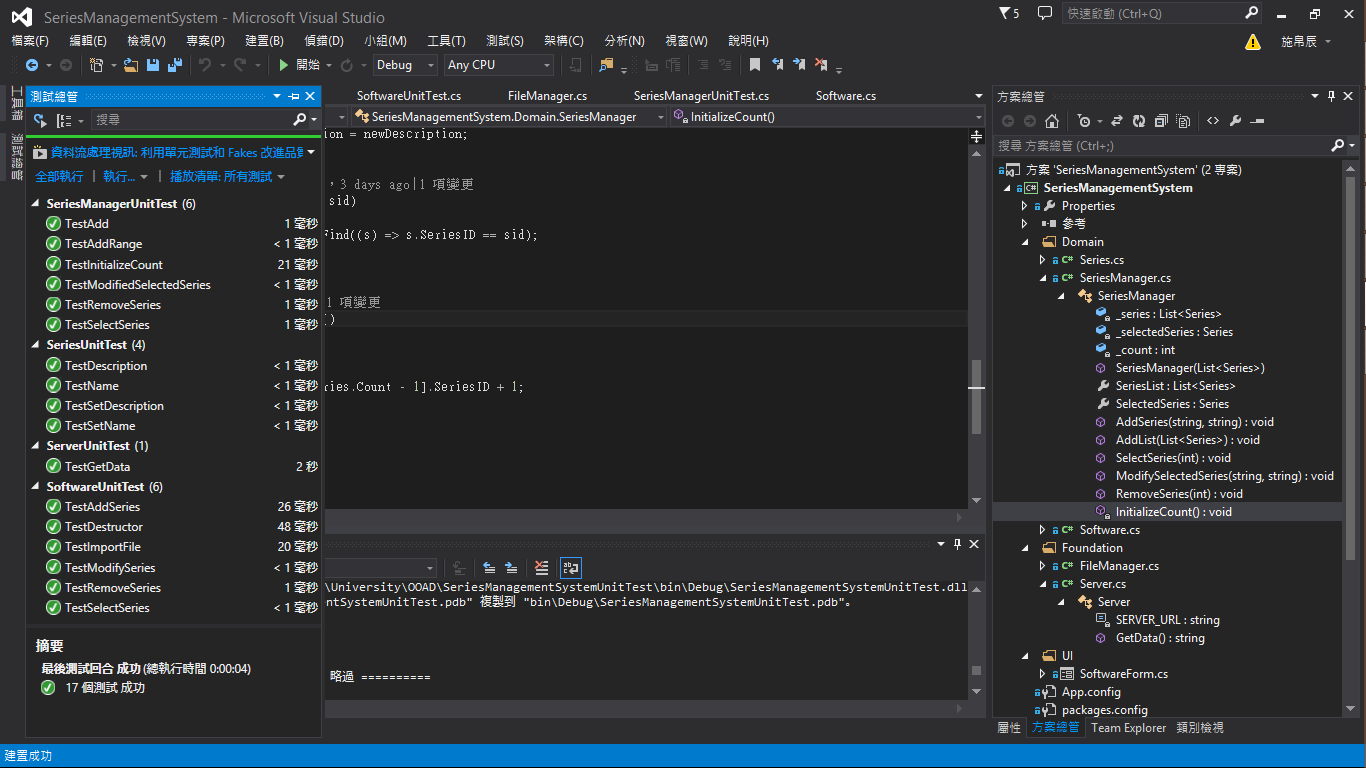
}

}

return data;

}

}

1. **Unit Testing**
   1. **Snapshot of Testing Result**
   2. **Unit Test Code Listing**
      1. **FileManager.cs**
      2. **SeriesManagerUnitTest.cs**

[TestClass]

public class SeriesManagerUnitTest

{

SeriesManager \_seriesManager;

Series[] \_series;

const int SeriesID = 0;

const string SeriesName = "manager's series";

const string SeriesDescription = "it is a series' description of manager";

const string ModifiedSeriesName = "modifiedSeries";

const string ModifiedSeriesDescription = "this is a modified description";

[TestInitialize]

public void Initialize()

{

\_seriesManager = new SeriesManager(new List<Series>());

\_series = new Series[3];

for (int i = 0; i < 3; i++)

{

\_series[i] = new Series(SeriesName + i.ToString(), SeriesDescription + i.ToString(), SeriesID + i);

}

}

[TestMethod]

public void TestInitializeCount()

{

PrivateObject privateObject = new PrivateObject(\_seriesManager);

Assert.AreEqual(0, privateObject.GetFieldOrProperty("\_count"));

privateObject.SetField("\_series", new List<Series>(\_series));

privateObject.Invoke("InitializeCount");

Assert.AreEqual(3, privateObject.GetFieldOrProperty("\_count"));

}

/// <summary>

/// Test function of AddSeries

/// </summary>

[TestMethod]

public void TestAdd()

{

// test AddSeries function with one parameter, series

\_seriesManager.AddSeries(\_series[0].Name, \_series[0].Description);

List<Series> seriesList = GetSeriesList();

Series series = seriesList.Last();

Assert.AreEqual(series.Name, \_series[0].Name);

Assert.AreEqual(series.Description, \_series[0].Description);

// test AddSeries function with two parameter, name ,and description

Initialize();

\_seriesManager.AddSeries(SeriesName, SeriesDescription);

seriesList = GetSeriesList();

series = seriesList.Last();

Assert.AreEqual(series.Name, SeriesName);

Assert.AreEqual(series.Description, SeriesDescription);

}

/// <summary>

/// test the function of AddRange with a parameter, List<Series>

/// </summary>

[TestMethod]

public void TestAddRange()

{

List<Series> content = GetSeriesList();

Assert.IsTrue(content.Count == 0, "the initialized series list is not empty");

List<Series> seriesList = new List<Series>(\_series);

\_seriesManager.AddList(seriesList);

content = GetSeriesList();

Assert.IsTrue(content.Count != 0, "the series list is still empty after adding a list of series");

}

/// <summary>

/// Test function of SelectSeries

/// </summary>

[TestMethod]

public void TestSelectSeries()

{

// add series into series manager

List<Series> seriesList = GetSeriesList();

seriesList.AddRange(new List<Series>(\_series));

// test initialization of selected series is empty

Assert.IsNull(\_seriesManager.SelectedSeries);

// test selected series after selecting the series

\_seriesManager.SelectSeries(2);

Assert.AreEqual(\_series[2], \_seriesManager.SelectedSeries);

\_seriesManager.SelectSeries(1);

Assert.AreEqual(\_series[1], \_seriesManager.SelectedSeries);

// test if manager does not find the series in the list, it returns null

\_seriesManager.SelectSeries(10);

Assert.IsNull(\_seriesManager.SelectedSeries);

}

[TestMethod]

public void TestModifiedSelectedSeries()

{

GetSeriesList().AddRange(new List<Series>(\_series));

\_seriesManager.SelectSeries(2);

Assert.AreEqual(\_series[2], \_seriesManager.SelectedSeries);

\_seriesManager.ModifySelectedSeries(ModifiedSeriesName, ModifiedSeriesDescription);

Assert.AreEqual(ModifiedSeriesName, \_seriesManager.SelectedSeries.Name);

Assert.AreEqual(ModifiedSeriesDescription, \_seriesManager.SelectedSeries.Description);

Assert.AreEqual(ModifiedSeriesName, GetSeriesList().Find((x)=>x.SeriesID==2).Name);

Assert.AreEqual(ModifiedSeriesDescription, GetSeriesList().Find((x)=>x.SeriesID==2).Description);

}

[TestMethod]

public void TestRemoveSeries()

{

List<Series> seriesList = GetSeriesList();

seriesList.AddRange(new List<Series>(\_series));

Assert.AreEqual(3, seriesList.Count);

\_seriesManager.RemoveSeries(1);

Assert.AreEqual(2, seriesList.Count);

Assert.AreEqual(-1, seriesList.IndexOf(\_series[1]));

}

/// <summary>

/// get the series list of series manager

/// </summary>

/// <returns></returns>

private List<Series> GetSeriesList()

{

return \_seriesManager.SeriesList;

}

}

* + 1. **SeriesUnitTest.cs**

[TestClass]

public class SeriesUnitTest

{

Series \_series;

const int SeriesID = 10;

const string SeriesName = "testSeries";

const string SeriesDescription = "this is a test Series Description";

const string ModifiedSeriesName = "modifiedSeries";

const string ModifiedSeriesDescription = "this is a modified description";

[TestInitialize]

public void Initialize()

{

\_series = new Series(SeriesName, SeriesDescription);

}

[TestMethod]

public void TestName()

{

Assert.AreEqual(SeriesName, \_series.Name);

}

[TestMethod]

public void TestDescription()

{

Assert.AreEqual(SeriesDescription, \_series.Description);

}

[TestMethod]

public void TestSetName()

{

Assert.AreEqual(SeriesName, \_series.Name);

\_series.Name = ModifiedSeriesName;

Assert.AreEqual(ModifiedSeriesName, \_series.Name);

}

[TestMethod]

public void TestSetDescription()

{

Assert.AreEqual(SeriesDescription, \_series.Description);

\_series.Description = ModifiedSeriesDescription;

Assert.AreEqual(ModifiedSeriesDescription, \_series.Description);

}

}

* + 1. **ServerUnitTest.cs**

[TestClass]

public class ServerUnitTest

{

Server \_server;

[TestInitialize]

public void Initialize()

{

\_server = new Server();

}

[TestMethod]

public void TestGetData()

{

string data;

data = \_server.GetData();

List<Series> series = JsonConvert.DeserializeObject<List<Series>>(data);

Assert.IsTrue(series.Count > 0);

}

}

* + 1. **SoftwareUnitTest.cs**

[TestClass]

public class SoftwareUnitTest

{

Software \_software;

PrivateObject \_privateObject;

Series[] \_series;

const String \_filePath = "../test.txt";

const string SeriesName = "Test Series";

const string SeriesDescription = "This is a test description";

const string ModifiedSeriesName = "modifiedSeries";

const string ModifiedSeriesDescription = "this is a modified description";

const string FILE\_PATH = "./dat/data.dat";

[TestInitialize()]

public void TestInitialize()

{

\_software = new Software();

\_privateObject = new PrivateObject(\_software, new PrivateType(typeof(Software)));

for (int i = 0; i < 3; i++)

{

\_software.AddSeries(SeriesName + i.ToString(), SeriesDescription + i.ToString());

}

}

[TestCleanup]

public void CleanUp()

{

if (File.Exists(FILE\_PATH))

File.Delete(FILE\_PATH);

}

[TestMethod]

public void TestAddSeries()

{

String name = "First Movie";

String description = "The first movie in the world.";

\_software.AddSeries(name, description);

Series s = GetLastSeries();

Assert.AreEqual(name, s.Name);

Assert.AreEqual(description, s.Description);

}

[TestMethod]

public void TestImportFile()

{

String name = "First Movie";

String description = "The first movie in the world.";

int seriesID = 1;

String fileContext = "[{ \"Name\":\"" + name + "\", \"Description\":\"" + description + "\", \"SeriesID\":"+seriesID+"}]";

PrepareImportFile(fileContext);

\_software.ImportFile(\_filePath);

Series s = GetLastSeries();

Assert.AreEqual(name, s.Name);

Assert.AreEqual(description, s.Description);

}

[TestMethod]

public void TestSelectSeries()

{

\_software.SelectSeries(1);

Assert.AreEqual(SeriesName + 1, GetSeriesManager().SelectedSeries.Name);

Assert.AreEqual(SeriesDescription + 1, GetSeriesManager().SelectedSeries.Description);

}

[TestMethod]

public void TestModifySeries()

{

\_software.SelectSeries(1);

Assert.AreEqual(SeriesName + 1, GetSeriesManager().SelectedSeries.Name);

Assert.AreEqual(SeriesDescription + 1, GetSeriesManager().SelectedSeries.Description);

\_software.ModifySeries(ModifiedSeriesName, ModifiedSeriesDescription);

Assert.AreEqual(ModifiedSeriesName, GetSeriesManager().SelectedSeries.Name);

Assert.AreEqual(ModifiedSeriesDescription, GetSeriesManager().SelectedSeries.Description);

}

[TestMethod]

public void TestRemoveSeries()

{

SeriesManager seriesManager = GetSeriesManager();

List<Series> seriesList = seriesManager.SeriesList;

Assert.AreEqual(3, seriesList.Count);

\_software.RemoveSeries(1);

Assert.AreEqual(2, seriesList.Count);

Assert.IsNull(seriesList.Find((s) => s.Name == SeriesName + 1));

}

[TestMethod]

public void TestDestructor()

{

Assert.IsFalse(File.Exists(FILE\_PATH));

\_software = null;

\_privateObject = null;

GC.Collect();

GC.WaitForPendingFinalizers();

Assert.IsTrue(File.Exists(FILE\_PATH));

}

private void PrepareImportFile(string fileContext)

{

if (File.Exists(\_filePath))

File.Delete(\_filePath);

using (FileStream fs = File.OpenWrite(\_filePath))

{

Byte[] info = new UTF8Encoding(true).GetBytes(fileContext);

fs.Write(info, 0, info.Length);

}

}

#region Get Private Object

private Series GetLastSeries()

{

SeriesManager seriesManager = GetSeriesManager();

Assert.IsNotNull(seriesManager.SeriesList);

Assert.IsTrue(seriesManager.SeriesList.Count > 0, "No any series in the list!");

return seriesManager.SeriesList[seriesManager.SeriesList.Count - 1];

}

private SeriesManager GetSeriesManager()

{

return \_privateObject.GetField("\_seriesManager") as SeriesManager;

}

#endregion

}

**Measurement**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **101820302 施帛辰** | | **101820340 鄒令業** | | **備註** |
| **HW #1** | | | | |
| 16/02/23 14:10~15:15 | **65 min** | 16/02/23 14:10~15:15 | **65 min** | **Meeting** |
|  |  | 16/02/28 13:30~14:00 | **30 min** | **Doc. Writing** |
| **Total** | **65 min** | **Total** | **95 min** |  |
| **HW #2** | | | | |
| 16/03/10 10:10~12:10 | **120 min** | 16/03/10 10:10~12:10 | **120 min** | **Meeting & Discussion** |
| 16/03/15 14:15~17:05 | **170 min** | 16/03/15 14:15~17:05 | **170 min** | **Meeting & Discussion** |
| 16/03/17 10:10~11:30 | **80 min** | 16/03/17 10:10~11:30 | **80 min** | **Meeting & Discussion** |
| **Total** | **370 min** | **Total** | **370 min** |  |
| **HW #3** | | | | |
| 16/03/29 14:10~15:30 | **80 min** | 16/03/29 14:10~15:30 | **80 min** | **Meeting** |
| **Total** | **80 min** | **Total** | **80 min** |  |
| **HW #4** | | | | |
| 16/04/26 14:20~17:10 | **170 min** | 16/04/26 14:20~17:10 | **170 min** | **Meeting** |
|  |  | 16/04/27 13:00~14:00 | **60 min** | **Coding** |
| 16/04/27 16:10~17:20 | **70 min** | 16/04/27 16:10~17:20 | **70 min** | **Meeting** |
| **Total** | **240 min** | **Total** | **300 min** |  |
| **HW #5** | | | | |
| 16/05/01  19:00~20:30 | **90 min** | 16/05/02  12:00~12:30 | **30 min** | **Coding** |
| 16/05/03  14:00~17:00 | **180 min** | 16/05/03  14:00~17:00 | **180 min** | **Meeting** |
|  |  | 16/05/04  11:00~11:30 | **30 min** | **Coding** |
| **Total** | **270 min** | **Total** | **240 min** |  |