# Final

> 這份考試主要由國立臺北科技大學 109 資工系黃漢軒所命題。

## 檔案架構

**請清掉前一次的作業**，確認提交是否符合以下的檔案架構，否則無法進行評分。

```
.
├── CMakeLists.txt
├── LICENSE
├── README.md
├── files.cmake
├── include
│   ├── CalculateHelper.hpp
│   ├── DieselEnergy.hpp
│   ├── ElectricEnergy.hpp
│   ├── EnergyInfo.hpp
│   ├── ExpressTrain.hpp
│   ├── ICalculable.hpp
│   ├── IReservable.hpp
│   ├── LocalTrain.hpp
│   ├── ReserveInfo.hpp
│   ├── ReserveTicketInfo.hpp
│   ├── TicketInfo.hpp
│   ├── TouristTrain.hpp
│   ├── Train.hpp
│   ├── TrainInfo.hpp
│   └── TrainSystem.hpp
├── scripts
│   ├── CodeCoverage.cmake
│   └── coverage.sh
├── src
│   ├── DieselEnergy.cpp
│   ├── ElectricEnergy.cpp
│   ├── EnergyInfo.cpp
│   ├── ExpressTrain.cpp
│   ├── LocalTrain.cpp
│   ├── ReserveInfo.cpp
│   ├── ReserveTicketInfo.cpp
│   ├── TicketInfo.cpp
│   ├── TouristTrain.cpp
│   ├── Train.cpp
│   └── TrainSystem.cpp
└── test
    ├── ut_CheckPoint1.cpp
```

```
    ├── ut_CheckPoint2.cpp
    ├── ut_CheckPoint3.cpp
    ├── ut_CheckPoint4.cpp
    ├── ut_CheckPoint5.cpp
    └── ut_TrainInfo.cpp
```

# 題目敘述

> 我們將嘗試描述一個火車系統（TrainSystem）。

對於臺灣的火車來說，我們可以分成快捷火車（Express Train，像是普悠瑪或者太魯閣）、區間車（Local Train）與環島列車（Tourist Train）

其中快捷火車與環島列車的訂票為對號座，區間車則不需要對號座。

對於快捷火車、區間車與環島列車，我們可以確保他們能夠用以下的兩種資訊來組成火車：

- 他們都會有火車資訊（TrainInfo）
    - 用來描述火車的車廂數量（Train）、座位數量（Seat）、每站票價（PricePerStation）、名稱（Name）與站牌（Station）。
- 他們都會有能源資訊（EnergyInfo）
    - 用來描述火車是使用柴油來當作能源（DieselEnergy）或者使用電力來當作能源（ElectricEnergy）。
    - 可以透過一些複雜的公式來算出這些能源可以跑多遠與跑多久（見 CalculateHelper）

對於快捷火車與環島火車，他們必須要可以提供對號座售票（Reserve）的服務，並且發售對號座車票（ReserveTicketInfo）

- 包含起站（StartStation）與迄站（EndStation）。
- 包含對號座資訊，也就是車次（Train）與座位（Seat）。
- 包含票價（Price，經過的站數乘以每站的票價）。
- 包含車次資訊（TrainInfo）。

對於區間車，他們必須要可以提供售票（GetTicket）的服務，並且發售車票（TicketInfo）

- 包含票價（Price，為一站的票價）。
- 僅包含車次資訊（TrainInfo）。

在我們能夠成功描述這些火車後，我們有一個火車系統（TrainSystem），可以進行以下的事情：

- 初始化多台火車到這個火車系統上。

- 取得某一台火車，對於這台火車能夠進行訂票等操作。
- 計算現在所有火車能夠行駛的距離總和。
- 計算現在所有火車能夠行駛的時間總和。

# 題目任務

## 第一階段：實作能源相關類別

**關鍵字：繼承類別、介面**

- 實作 `EnergyInfo` 類別。
  - 具有建構子 `EnergyInfo(int energy)` 用於初始化能源值。
  - 具有 Getter `int GetEnergy()` 用於取得能源值。
  - 實作 `ICalculatable` 介面，並將 Pure Virtual Function 再次指派給子類實作。
- 實作 `DieselEnergy` 類別。
  - 繼承 `EnergyInfo` 類別。
  - 具有建構子 `DieselEnergy(int energy)` 用於初始化能源值至父類。
  - 實作 `int GetDistance()` 用於取得該柴油能源可以讓火車跑多遠。
    - 你可以使用 `CalculateHelper` 中的函數來幫助你完成。
  - 實作 `int GetArrivalTime()` 用於取得該柴油能源可以讓火車跑多久。
    - 你可以使用 `CalculateHelper` 中的函數來幫助你完成。
- 實作 `ElectricEnergy` 類別。
  - 繼承 `EnergyInfo` 類別。
  - 具有建構子 `ElectricEnergy(int energy)` 用於初始化能源值至父類。
  - 實作 `int GetDistance()` 用於取得該電力能源可以讓火車跑多遠。
    - 你可以使用 `CalculateHelper` 中的函數來幫助你完成。
  - 實作 `int GetTerminalStationArrivalTimeInSecond()` 用於取得該電力能源可以讓火車跑多久。
    - 你可以使用 `CalculateHelper` 中的函數來幫助你完成。

## 第二階段：實作車票相關類別

- 實作 `TicketInfo` 類別，該類別為一般車票。
  - 具有建構子 `TicketInfo(std::shared_ptr<TrainInfo> trainInfo)` 用於初始化車票資訊。
  - 具有 Getter `std::shared_ptr<TrainInfo> GetTrainInfo()` 用於取得火車相關資訊。
  - 具有 `virtual int GetPrice()` 用於初始化價格，對於普通車票來說，無論搭多少站，均為一站的車票錢。
- 實作 `ReverseInfo` 類別，該類別用於描述對號座之相關資訊。

    - 具有一般建構子。
    - 具有建構子 `ReserveInfo(int train, int seat)` 用於初始化對號座之資訊，為第幾車與第幾個座號。
    - 具有 Getter `int GetTrain()` 取得車次資訊。
    - 具有 Getter `int GetSeat()` 取得座號資訊。
- 實作 `ReverseTicketInfo` 類別，該類別為對號座車票，為一般車票的擴充。
    - 繼承 `TicketInfo`。
    - 具有建構子 `ReserveTicketInfo(int startStationIndex, int endStationIndex, std::shared_ptr<TrainInfo> trainInfo)` 用於初始化車票資訊與起站、迄站索引值。
    - 具有 Getter `std::string GetStartStation()` 取得起站的車站名稱。
    - 具有 Getter `std::string GetEndStation()` 取得迄站的車站名稱。
    - 覆寫 `int GetPrice()`，對於對號座的車票計價，以通過的車站數量乘以每站的車票價格。
    - 具有 Getter `ReserveInfo GetReserveInfo()` 取得對號座資訊，由 `TrainInfo` 中的 `ReserveSeat()` 來產生對號座資訊 `ReserveInfo`。

## 第三階段：使用組合完成 `Train` 類別

**關鍵字：組合**

- 讓 Train 具有建構子 `Train(std::shared_ptr<TrainInfo> trainInfo, std::shared_ptr<EnergyInfo> energyInfo)`
    - 用於初始化火車資訊與能源資訊。
    - 須為智慧指標傳入。
- 讓 Train 具有 Getter `std::shared_ptr<TrainInfo> GetTrainInfo()`
    - 用於取得火車的資訊。
- 讓 Train 具有 Getter `std::shared_ptr<EnergyInfo> GetEnergyInfo()`
    - 用於取得火車的能源資訊。

## 第四階段：完成 `ExpressTrain`、`LocalTrain` 與 `TouristTrain` 類別

**關鍵字：Dependency Injection、介面**

- 對於 `ExpressTrain`，繼承 `Train` 與實作 `IReservable` 介面。
    - 實作傳入火車資訊與能源資訊的建構子 `ExpressTrain(std::shared_ptr<TrainInfo> trainInfo, std::shared_ptr<EnergyInfo> energyInfo)` ，並使用 Initialize List 初始化到父類上。
    - 實作 `Reserve()` 函數，用於模擬售出對號座，回傳 `ReserveTicketInfo`。
        - 票價為經過的站數乘以一站的票價。
- 對於 `TouristTrain`，繼承 `Train` 與實作 `IReservable` 介面。

    - 實作傳入火車資訊與能源資訊的建構子 `TouristTrain(std::shared_ptr<TrainInfo> trainInfo, std::shared_ptr<EnergyInfo> energyInfo)`，並使用 Initialize List 初始化到父類上。
    - 實作 `Reserve()` 函數，用於模擬售出對號座，回傳 `ReserveTicketInfo`。
        - 票價為經過的站數乘以一站的票價。
- 對於 `LocalTrain`，僅繼承 `Train`。
    - 實作傳入火車資訊與能源資訊的建構子 `LocalTrain(std::shared_ptr<TrainInfo> trainInfo, std::shared_ptr<EnergyInfo> energyInfo)`，並使用 initialize_list 初始化到父類上。
    - 實作 `GetTicket()` 函數，用於模擬售出車票，並回傳 `TicketInfo`。
        - 票價為一站的票價。

## 第五階段：完成 `TrainSystem`

**關鍵字：多型**

- 實作 `TrainSystem(std::vector<std::shared_ptr<Train>> trains)`
    - 初始化多台火車到這個火車系統上。
    - 實作 `std::shared_ptr<Train> GetTrain(int index)`
    - 取得某一台火車，對於這台火車能夠進行訂票等操作。
- 實作 `int GetTotalDistance()`
    - 計算現在所有火車能夠行駛的距離總和。
- 實作 `int GetTotalArrivalTime()`
    - 計算現在所有火車能夠行駛的時間總和。
- 實作 `int GetSize()`
    - 計算現在的火車數量。

# 題目備註

- 你不需要額外撰寫測試，不用達到 95% 的覆蓋率。
- 我們有提供給你測試，可自行開啟它來進行測試。

# 任務配分

- 依照測試案例進行配分
    - [\20%\] 實作 `EnergyInfo`、`DieselEnergy`、`ElectricEnergy` 類別。
    - [\20%\] 實作 `TicketInfo`、`ReserveInfo`、`ReserveSeatInfo` 類別。
    - [\20%\] 實作 `Train`、`TrainInfo` 類別。
    - [\20%\] 實作 `ExpressTrain`、`TouristTrain`、`LocalTrain` 類別。

- [\20%\] 實作 `TrainSystem` 類別。

# 注意事項

- 若你想要有空指標，請將指標賦值（assign） `nullptr` ，這個特殊的指標可用於指向不存在的東西。
- 對於 Google Test 的測試函數：
  - 使用 `ASSERT_EQ` 來測試不會有精度誤差的值（例如：整數、字串等）
  - 使用 `ASSERT_NEAR` 來測試具有精度的值
  - 使用 `ASSERT_THROW` 來測試該函數會不會拋出例外
  - 使用 `ASSERT_FALSE` 來測試該值是否為 `false`
- 你不應該上傳 `bin` 資料夾至專案上。
  - 你的功課不應該出現 Memory Leak，否則將會扣除作業總分 10 分。
  - 你不應該上傳 `/bin` 資料夾至專案庫，編譯結果不應該上傳至專案庫上，若在助教確認功課評分時 `/bin` 資料夾存在在專案庫中，扣除作業總分 5 分。
- 你可以使用 `debugger` 工具來進行除錯，有利於尋找漏洞與理解漏洞為何會發生。