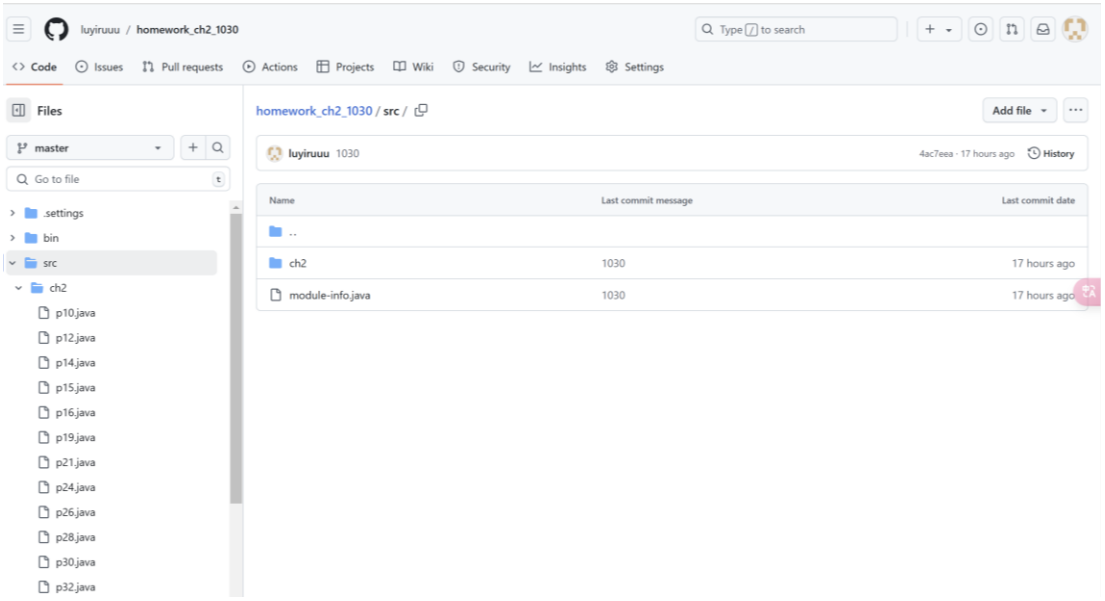
1. Git Hub 連結: https://github.com/luyiruuu/homework_ch2_1030.git
2. Git Hub 截圖:



3. 心得:

第 2 章我們主要學習了條件處理、迴圈敘述和陣列，是程式設計中最基本同時也是最重要的概念。條件處理包括條件式的表示法與關係運算子的應用，透過邏輯判斷讓程式執行不同操作。條件式是一個會回傳布林值的表達式，結果為‘true’或‘false’。我們在作業中較常用的關係運算子包括’==’ (等於)、’>‘(大於)、’<’(小於)、’>=’(大於等於)、’<=(小於等於)，來比較兩者的大小，或是控制迴圈的執行次數。if 敘述可以根據條件執行某段程式碼，需要注意如果包含多行程式碼則必須使用’{ }’ 將內容框住，避免邏輯錯誤。

此外，if-else 敘述可以在條件成立或不成立時執行不同的區塊，而 if-else if-else 敘述則可以依據多個條件做進一步的選擇。switch 敘述是一種多重選擇的敘述，可以根據一個表達式的值，選擇並執行對應的程式區塊。switch 在 case 敘述後要記得加上’break’，否則程式會繼續執行下一個 case，造成錯誤。我們也可以透過條件運算子’?:’進行條件判斷，讓程式更為簡潔。

迴圈敘述部分:可以重複執行某段程式碼，直到特定條件不成立為止。for 迴圈適用於已知執行次數的狀況。while 迴圈則會在條件成立時持續執行，適合用於執行次數不確定的情境。do-while 迴圈與 while 迴圈相似，但不同之處在於它會至少先執行一次程式內容，然後再判斷條件。巢狀迴圈則是指在一個迴圈內包含另一個迴圈，這種結構適合用於處理如多維陣列的處理。在迴圈中使用 break 敘述可以強制中斷當前的迴圈，跳出並執行迴圈之後的程式碼；continue 敘述則會暫停本次迴圈的執行，直接進入下一次迴圈操作。

最後是陣列部分:陣列是一種用於儲存多筆相同型態資料的資料結構，

具有「一次可存多個值」的優點。宣告一個一維陣列的方式為'int test[5]'，表示 'test'為包含 5 筆整數的陣列。陣列中的第一個元素的索引值為 0，因此 'test[0]'代表陣列中的第一筆資料。使用陣列可以大幅提高程式的可讀性與簡潔性，當儲存大量資料時，無需宣告多個變數來逐一儲存，而是透過索引值來取得資料。此外，可以利用排序方法，例如 Arrays 類別中的 sort 方法，可以輕鬆地將陣列按遞增排序，若需按遞減排序，則可搭配 Collections 類別中的 reverseOrder 方法反轉順序。多維陣列則是指陣列中包含多個列或欄，例如'int score[2][5]'，表示一個 2x5 的二維整數陣列。多維陣列適合用於表示矩陣等較複雜的結構，然而，使用陣列時應注意索引值不能超出陣列範圍，否則會發生讀取錯誤或例外情況。

4. Chatgpt:
   - 在 Java 中，巢狀迴圈的效率如何？巢狀迴圈層數增加時，會對程式效能帶來哪些影響？

     巢狀迴圈會根據層數成倍地增加運算次數，例如兩層巢狀迴圈若每層執行 100 次，則內部程式會執行 $100 \times 100 = 10,000$ 次。巢狀層數增加時，運算次數以指數級成長，導致程式效率降低，尤其在處理大量資料或需要高效能的情境下。過多層次的巢狀迴圈會增加 CPU 負擔，導致程式運行變慢，並可能影響記憶體的分配。因此，程式設計上應儘量減少巢狀迴圈層數，或使用更高效的演算法以提升效能。

   - 使用 break 和 continue 進行迴圈控制有哪些潛在的風險？應該在什麼樣的情境下避免使用這些指令？

     break 和 continue 會影響迴圈的正常流程控制。break 會直接結束整個迴圈，可能導致意外中止而無法完成所有操作；continue 則會跳過當前迴圈剩餘的程式碼，並進入下一次迴圈，若過度使用可能讓程式難以追蹤和維護。這些指令在簡單迴圈中較易控制，但在巢狀迴圈或複雜條件下容易導致錯誤或混淆，應避免在邏輯複雜的程式中過度使用，除非確定其行為能夠被清楚理解並控制。

   - 如果不使用 Java 的 Arrays 類別的情況下，如何實現陣列的排序？有哪些排序演算法適合？

     若不使用 Arrays 類別，可以自行實現排序演算法，如 Bubble Sort、Selection Sort、Insertion Sort、Quick Sort 和 Merge Sort。Bubble Sort 和 Insertion Sort 適合少量或近似排序的資料，因為這些演算法簡單但效率較低。Quick Sort 和 Merge Sort 適合大量無序的資料，因為這些演算法的時間複雜度較低，Quick Sort 平均時間複雜度為 $O(n \log n)$，適合一般用途，而 Merge Sort 的穩定性較高，適合需要保持相同元素順序的情況。選擇演算法時應考量資料量和是否需要穩定排序，以達到最佳效能。

   - if與switch敘述各有哪些適用的情境？在程式設計中應如何選擇使用？

if 敘述適合用於條件比較多樣且判斷式較為複雜的情境，如需進行多層次判斷或使用邏輯運算子（`&&`、`||` 等）時，使用 if 敘述能更靈活地控制程式流。而 switch 敘述則更適合處理單一變數的多重選擇情況，例如處理數字、字元或字串等特定值的匹配，因為它能提高程式的可讀性和執行效率。一般來說，當條件數量較多且固定時，如狀態機切換或根據選項選擇動作時，使用 switch 更合適，而需要動態條件或多條件判斷時則選用 if 敘述。

5. 讀書會
   - 組員:陸蕙如、羅文均、官奕愷
   - 討論時間:10/30　22:00～22:30(共計 30 分鐘)
   - 討論地點:線上(Line)
   - 照片: