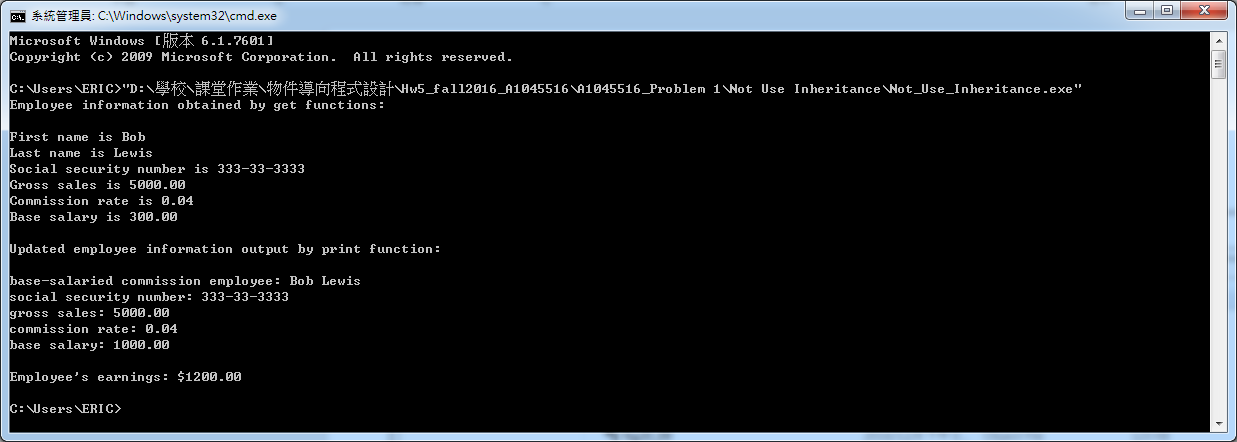
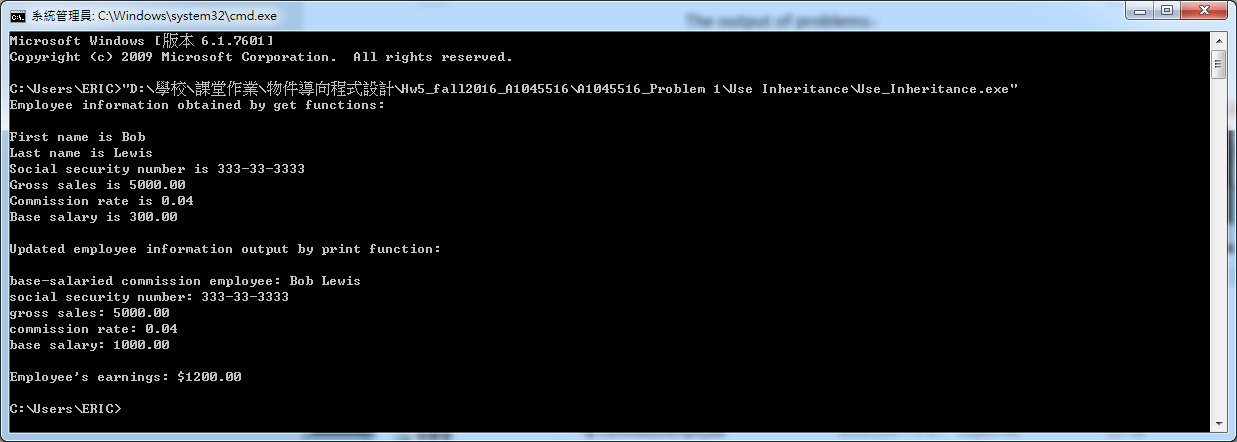
The output of problems

Problem 1:

Not Use Inheritance

Use Inheritance

在將其程式寫成Composition以及Inheritance兩種不同的形式之後，讓我發覺到了其寫成Inheritance的形式是有許多的好處的，就以寫成Composition的形式而言，可以發覺到因為是將兩個類別的內容給組合起來，所以其標頭檔裡頭所定義的東西十分的多，那麼這也意味著當我們在進行實作的時候，其實作的內容勢必會十分的龐大，那麼這也造成了一個問題的發生，就是其程式碼裡頭所寫的內容不易去檢查，並且也很難去區分出每一個member function出來，使得其對於程式之後的維護會十分的艱難。

那反觀使用Inheritance的形式，我們可以發覺到其每一個類別所定義的member function以及data member都清楚的區分在不同的類名下，並且當其另一個所要定義的member function或者是data member早已在另一個class當中就已經被定義過了，這時就可利用繼承的概念來去獲得該類的所有的member function以及data member，那麼藉由這樣我們就只需再去新增一些我們所需要的member function或者是data member就行了，而不用在去重新定義那些早已在前一個類就定義過的member function以及data member，這不但會使我們節省了許多的時間，並且也使得其程式碼更加的簡潔以及清晰，並且其實作的部份也只需再針對其所新增的member function或者是data member在去做一些編程的動作就行了，而不是像Composition的形式一樣必須再將其已定義過的以及尚未定義過的都必須再重新寫過，這不但會造成其時間上的浪費並且也會使得其程式的程式碼變得十分的亂，造成其日後要去做檢視、新增或者是修改的動作的時候，都將會變得十分的艱難。

所以從這邊我們就可以看出其寫成Inheritance的形式的好處分別如下:

1. 可以快速看出其類與類之間的關係
2. 使其程式碼變得更加簡潔，而不會產生任何冗餘的程式碼區段
3. 加快其編程的速度
4. 提高其日後程式的可維護性
5. 可以只針對單個類別做修改的動作並且不會去影響到其它類別，增加其編程的彈性

當然，其寫成Inheritance的形式的好處絕不單單只有上述的這些，一定還有許多的好處，而這也就是為什麼其繼承這個概念會在其物件導向程式設計當中占據有極其重要的地位，一旦沒了它的存在，將會對其編程的過程造成十分重大的影響，這也是為什麼我會比較頃向於寫成Inheritance的形式而不是寫成Composition的形式，就是基於以上的這些原因。

Problem 2:

