《系統分析與設計》

試題評析	本次命題第一題、第三題為系統開發題型，第二題屬專案管理，第四題則為UML之類別圖與物件導向程式碼之對映，也是課程一再提醒的重點。 整體而言，命題難易適中，學員如何在有限時間內盡量呈現自己的想法以及繪製範例圖形，將會是本科得分關鍵。
高分命中	第一題：《高點系統專案管理講義》第二回，張又中編撰，頁 2-35。 第二題：《高點系統專案管理講義》第九回，張又中編撰，頁 9-42； 《高點系統專案管理講義》第十回，張又中編撰，頁 10-23~24。 第三題：《高點系統專案管理講義》第二回，張又中編撰，頁 2-18~23； 《高點系統專案管理講義》第六回，張又中編撰，頁 6-21~22。 第四題：《高點系統專案管理講義》第六回，張又中編撰，頁 6-6~7； 《高點系統專案管理講義》第八回，張又中編撰，頁 8-15。

一、一般來說，建立一個新系統，主要有自行開發，使用套裝軟體，與外包三種方式。假設您負責規劃該系統的建置，請問要如何評估與擬定選擇方案的策略？（25 分）

【擬答】

考量因素	自行開發	使用套裝軟體	委外
機密性	資料處理流程與資料庫屬高度機密	需考量資料處理流程或是資料庫位於組織內/外部	系統故障或資料外洩不會引起嚴重問題
獨立性	系統彼此之間關聯性高	系統越獨立越適合，亦可購買整體解決方案	系統愈獨立則愈方便委外，成功率也愈高
企業策略	IT 基礎建設或其上的系統，提供了專屬的競爭優勢	同委外	IT 基礎建設或其上的系統支援組織運作，但不具策略優勢
核心能力	IT 的專業知識屬核心能力	同委外	IT 的專業知識非組織成功關鍵
成本效益	委外的成本高於自行開發	使用套裝軟體的成本明顯低於自行開發	委外的成本明顯低於自行開發
合作夥伴	缺乏可靠、有能力且有意願的合作夥伴	有可靠的套裝軟體供應商	有可靠、有能力且有意願的合作夥伴
套裝軟體	IT 基礎建設與其上的系統是獨一無二的	有合適的套裝軟體	有合適的解決方案
技術進步性	公司吸引並留下 IT 專才，並可在合理成本下採用新 IT	由套裝軟體開發廠商負責後續升級、改版	無法應付新 IT 進步的速度與複雜性
推行所需時間	有足夠的時間自行開發	節省開發時間，僅需導入與教育訓練	自行開發所需時間超過組織的要求甚多
快速的開發工具支援	有	無	無

資料來源：修改自《資訊管理理論與實務》(六版)，謝清佳、吳琮璠

二、專案管理時，要如何激勵成員達成專案目標？要避免那些激勵方式？如何擬定避免衝突的策略？（25 分）

【擬答】

激勵可以激發工作潛能、鼓舞工作士氣、誘導熱忱，其為一種人性化的管理，也是領導的有效法則之一。激勵必須符合員工的期待、滿足員工的需要與偏好，且能反映員工努力的貢獻，同時經得起員工間的比較，讓員工覺得不但合理且公平，才能提高工作意願與士氣。

Herzberg 雙因子理論		Maslow 需求層次	組織因素
激勵因子	成就、升遷、責任、認同感、工作挑戰、個人專業與成長	自我實現	晉升、挑戰性、創造力、工作成就
		自尊心與榮譽感	責任感、工作本身、工作頭銜、績效報償、同事主管讚賞
保健因子	薪資、個人生活、工作地位、工作保障、上級領導、工作環境、政策與行政、與上級關係、同事關係、部屬關係	愛與社會活動	同事友誼、監督地位、和諧工作團體
		安全感	薪資福利、工作保障、安全工作環境
		基本生理	基本薪資、工作環境、空調、停車位、自助餐

資料來源：修改自《軟體專案管理》，林信惠等

避免衝突的策略有：

1.對事不對人
當人受到抨擊或是威脅時，其會直覺地進行自我防禦且不會去解決問題。此時應聚焦於問題本身而非人身攻擊。

2.針對利益而非立場
當人專注於立場時，往往變成意志力的對抗，因而造成協調的拖延。可藉由真誠的瞭解別人來滿足其需求與利益。

3.為共同利益獻策
在利益明確且共同探究彼此的策略下，創造出合乎利益的機會。可利用合作式的腦力激盪來達到雙贏的局面。

4.可能時以客觀標準為本
遵循所發展出的標準和規則來協助處理所爭論的部分。

5.處理非理智的成員
當遭遇到觀念上有很強的輸贏看法的人時，將會使事情變得難以處理，可使用柔性的協商談判法。

三、請詳述並比較下列三種反覆式的軟體開發方式之特性：使用案例驅動開發、功能驅動開發與測試驅動開發。良好軟體的開發通常在開發循環的不同階段，會使用以上的那些開發模型？（25 分）

【擬答】

測試驅動開發(Test Driven Development, TDD)倡導先寫測試程式，而後編碼實現其功能而得。測試驅動開發始於 1990 年代，其目的為取得快速回饋並使用「Illustrate the Main Line」的方法來構建程式。

測試驅動開發先實現功能，在測試的輔助下，快速實現其功能；接下來再重構，在測試的保護下去除冗餘的程式碼，提高程式品質。測試驅動整個開發過程：首先，驅動程式碼的設計和功能的實現；其後，驅動代碼的重構與再設計。

使用案例驅動開發(Use Case Driven Development, UDD)是以使用案例為核心，其描述某些角色利用系統完成某事件的流程順序(Jacobson, 1992)，亦即使用者使用系統的實例(Case)，是為物件導向分析與設計的核心，亦為 UML 的基礎，可衍生其他 UML 圖形。

使用案例驅動開發在系統規劃階段，可協助釐清系統範疇，並可於需求分析階段，捕捉系統需求與流程。此外，於系統開發完成後，亦可做為系統功能驗證與測試的依據。

功能驅動開發(Feature Driven Development, FDD)為一模型驅動、目的導向的快速反覆開發過程，其以架構(Architecture)為中心，強調功能驅動、快速反覆運算，既能保證快速開發，又能保證適當系統品質與系統文件，非常適合中小型團隊開發管理，也適用於需求經常變動的專案。

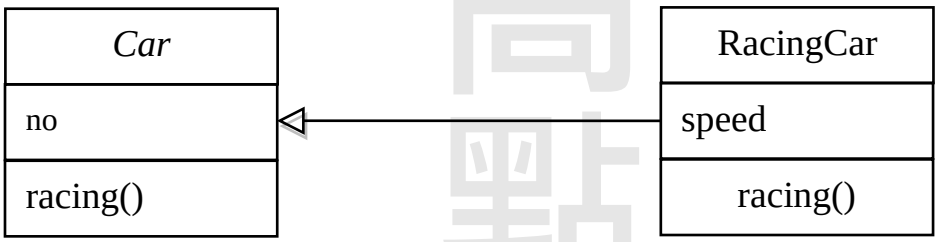
功能驅動開發首先對整個專案建立一整體模型，然後透過兩週內一系列"Design by Feature, Build by Feature"的反覆

運算，逐步豐富模型功能內容。在開發過程中，開發計畫的制定、報告的生成、進度的追蹤均是以功能為單位進行。

四、請說明抽象類別（abstract class）與其子類別分別與行為（method）之間的關係。請舉一個使用抽象類別的例子，畫出其類別圖與其相對應的程式碼。（25 分）

【擬答】

抽象類別為擁有抽象操作(Abstract Operation)的類別，僅宣告方法(Method)名稱而不實作當中的邏輯。其為未定義完全的類別，所以不能直接建立物件，只能被擴充，並於擴充後完成未完成的抽象方法定義之。




```
abstract class Car{
        int no;
        void racing();
}

class RacingCar extends Car{
        double speed;
        void racing(){
                System.out.println("Now speed is"+speed);
        }
}
```