Systems Analysis and Design 52.234

# Outstanding Coursework 2004-2005

## General instructions and marking

The outstanding coursework is for those students who took the class in 2004-2005 and who have not yet cleared the coursework for the class. It consists of two parts: a theoretical part and a practical part. Each part is worth 50 marks giving an overall total of 100 marks. In order to clear the coursework you need to get **at least 20 marks in each part** and an **overall mark of at least 60.**

The deadline for the submission of this coursework is **2pm Friday, 11th August, 2006**.

## Submission guidelines

1. Your submission should include a cover page including your name, your registration number, and a signed statement stating that the work submitted is your own.
2. Your submission should be taken to the departmental office L10 no later than 14:00 on the submission deadline date.
3. All submission material should be typed. All UML diagrams should be created using Together Architect. All other diagrams should be created using an appropriate software package. Any handwritten material will not be marked.

**Note:** Failure to comply with the submission guidelines will lead to a reduction of marks.

## Part A: Theory (50 marks)

1. Describe three reasons why large software systems are often of poor quality, late, and overbudget. (**3 marks**)
2. System planning techniques include SWOT (Strengths, Weaknesses, Opportunities, Threats), VCM (Value Chain Model), BPR (Business Process Re-engineering) and ISA (Information Systems Architecture). For what purposes are these techniques used? Compare and contrast two of these approaches. (**7 marks**)
3. Software metrics can be used to measure two aspects of software: the quality of the software product; the quality of the software development process. For each aspect, describe one metric that can be used. how might these metrics be used to influence project management? (**4 marks**)
4. In order to assess the technical feasibility of a project it is necessary to carry out a technical risk analysis. What are the three aspects that determine the technical risk of a project? How does each aspect relate to the risk estimation for the project? (**6 marks**)
5. What criteria might be used to select an appropriate mix of requirements elicitation techniques for a particular project? (**7 marks**)
6. Describe in detail each of the stages of the software lifecycle. (**6 marks**)
7. In which way is inheritance different to generalization? What is the difference between interface and implementation inheritance? Are there any problems associated with interface and implementation inheritance? (**4 marks**)
8. What is meant by coupling and cohesion? Describe two forms of coupling and two forms of cohesion, giving examples. (**5 marks**)
9. Describe each of the components in the Model-View-Controller architecture? What are the advantages offered by this architecture? (**5 marks**)
10. What is the purpose of a design pattern. Name and briefly describe two design patterns. (**3 marks**)

## Part B: Practical (50 marks)

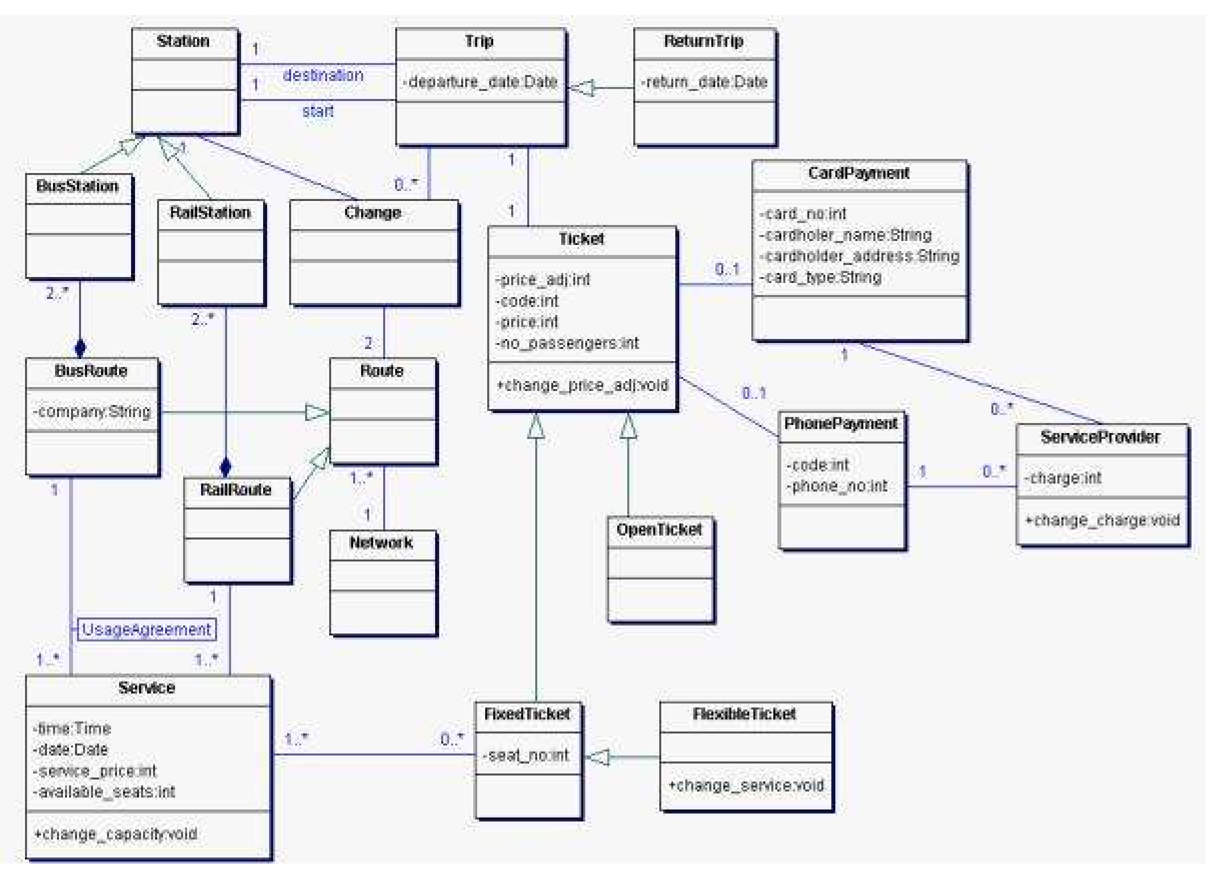
In this part you will continue to develop the design of the information system for Stinking Trains S.A. (see the details about the class project on the class website).

Stinking Trains S.A. has decided it is crucial for its operations that the new information system should allow members of its finance department to view information regarding the sales of tickets and the overall revenue collected on its various services. As a member of the development team for the new system you have been assigned the task of designing the user interface for the "`View Overall Revenue`" use case. Your design should include the following deliverables:

1. **Use Case Description**. Provide an extended real use case description for the `View Overall Revenue` use case. The use case allows staff of the finance department to select a particular period of time and to see what the overall revenue was for the rail company from all tickets sold during this period of time. The calculation of the revenue must take into account that customers have not necessarily paid the full service price when purchasing their tickets (e.g. it is common for tickets involving a large number of services or passengers to be charged at a discounted price usually referred to as a price adjustment). Moreover, the calculation should also take into account that a part of the final ticket price does not come back into the rail company's accounts. There are two reasons why this might happen. First, all ticket payments by credit/debit cards and phones involve a certain service charge. Second, all tickets involving bus services (i.e. services carried out by bus companies) require that the bus companies involved receive a certain amount in accordance to the usage agreements signed between the bus and the rail companies. Finally, the execution of the use case should detail to the staff of the finance department all the above details (i.e. the overall amount of money collected by the rail company, the overall revenue of the rail company, the overall charges paid to each service provider and the overall amount paid to each bus company, etc.).
2. **Activity Diagram**. Provide an activity diagram for the `View Overall Revenue` use case, according to the description you provided above.
3. **Overall Revenue Interface Screen Layout.** Design the layout of the main screen and any secondary screens that support the `View Overall Revenue` use case, according to the description you provided above.
4. **Sequence Diagram**. Provide a sequence diagram for the `View Overall Revenue` use case. Your diagram should include both boundary (user interface) and entity classes and should describe the interaction between them during the execution of the use case (see also the class diagram provided below).
5. **Statechart**. Provide a statechart describing the dynamic behaviour of each element of the user interface for the `View Overall Revenue` use case as designed above.

**Supporting Material, Assumptions and Advice**

- The consistency between the various deliverables is of paramount importance.
- Your user interface design should be based on the following class diagram:



- Your screen layout design descriptions should consist of two parts: (a) a picture depicting the various user interface elements that make up the screen, (b) some text explaining the role of each user interface element. Note that the text should not be excessive (no more than a page).
- You can use the documentation of the java.awt package as your source of reference for available user interface elements. Note that if you use element appearing in the java.awt package, then you don't have to explain how they work.