The University of Melbourne
Department of Computing and Information Systems
SWEN30006 Software Modelling and Design
Semester 1 Exam 2016

Reading Time: 15 minutes
Writing Time: 2 hours
Total marks for this paper: 120

This paper has 8 pages including this page.

Authorised materials: No materials are authorised.

Instructions to invigilators:
Each student should initially receive a script book.

Instructions to Students:
- Answer all questions.
- The marks for each question are indicated at the beginning of each question.
- The marks are an indication of how much time should be spent on the question.
- *Clearly number* each question.
- Make sure that you add your student number to each answer script.

This examination contributes 60% of your total marks for the subject.

This page intentionally left blank.

Question 1. [9 marks]

Briefly explain each of the following terms:

1. Actor
2. Scenario
3. Use Case

Question 2. [9 marks]

Consider the following passage.

> A museum plans to install tablet computers throughout for use by visitors. Initially the tablets will (a) show visitors information about the adjacent exhibit (presented section by section with the option to skip a section at any stage), with the possibility of a search for exhibits related to the current on-screen topic (b) let visitors browse current topics of interest from the media, with the possibility of a search for exhibits related to the current on-screen topic (c) allow visitors to request urgent assistance from museum staff.

Ignoring the issue of how information is placed in the system, model this passage in a use case diagram.

Question 3. [6 marks]

"There is no point commencing design until we have a complete set of use cases, a complete domain model and a full set of system sequence diagrams".

Is this ever a valid position to take? What is the alternative? Discuss.

Question 4. [24 marks]

This question relates to the following domain model. This model is similar but not identical to the model provided to you for project part B.
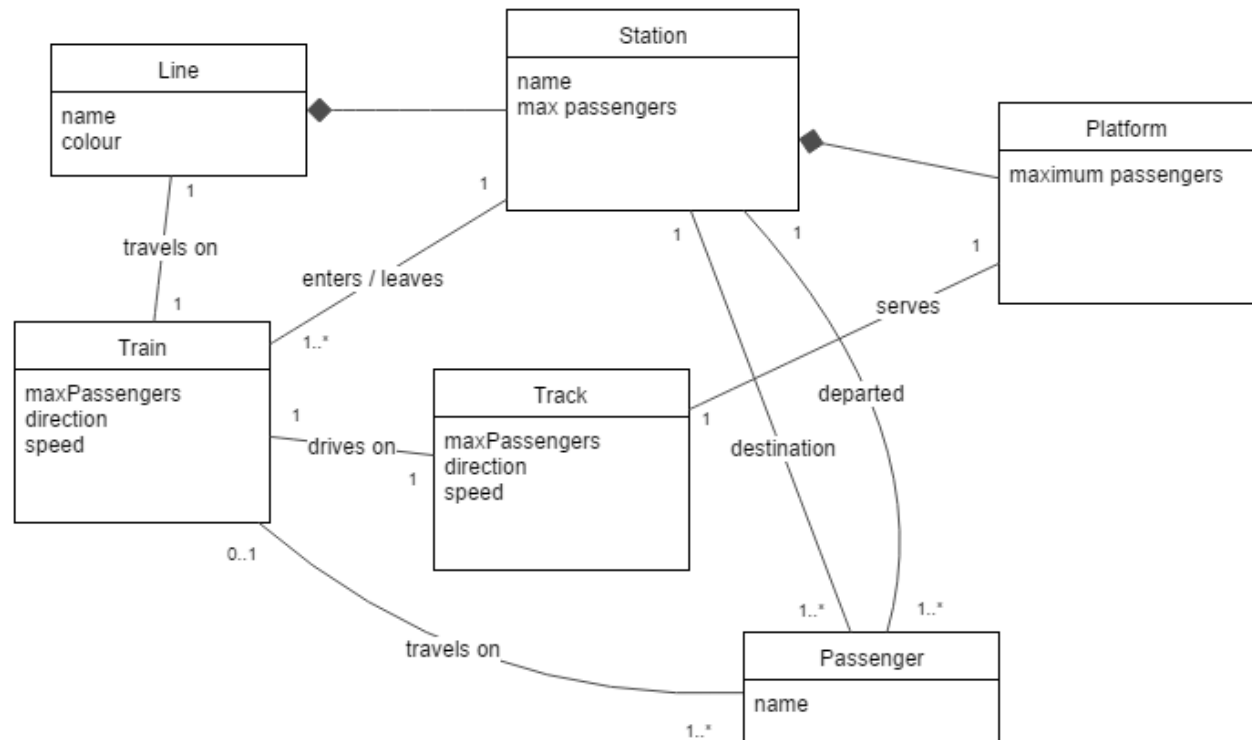
Question 4 Part 1. [9 marks]

Please answer true or false to each of the following, providing a brief justification for your answer:

a) For every passenger travelling on a train, a unique line colour can be determined corresponding to the current line on which they are travelling.
b) If a platform in the real world is the only platform at its station, this model allows that platform to be represented without its corresponding station.
c) For every passenger who departed a particular station, we can always determine a direction of travel for them.
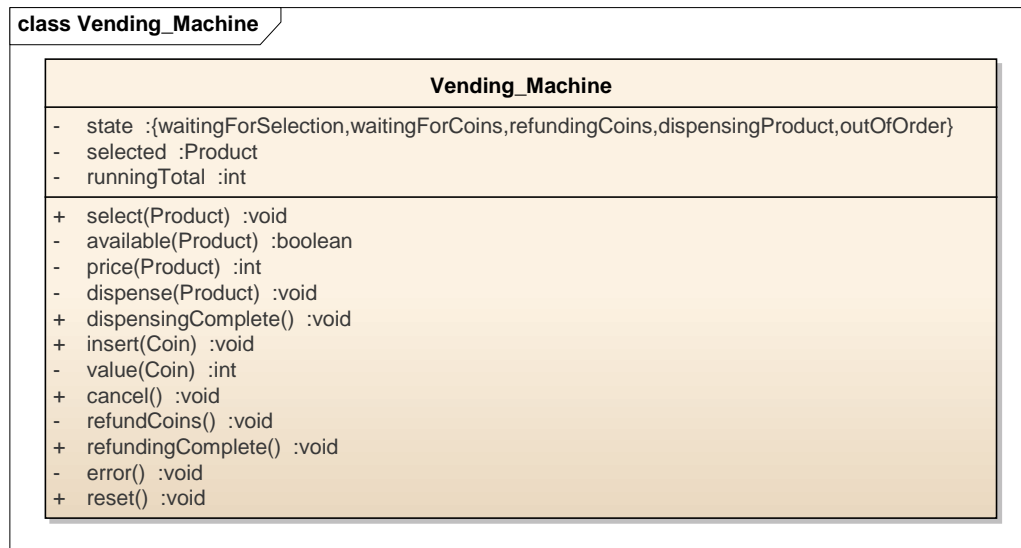
Question 4 Part 2. [15 marks]

Please draw extensions to the domain model meeting the requirements in the text box below. Only redraw elements (no need to include attributes if unchanged) from the original model if they are directly connected to new elements in the extension.

The Rail department wants to consider the impact of road crossings on the rail network. They want to have information about roads including the road name (e.g. Swanston) and the suffix (e.g. road, street, avenue), the road category (e.g. freeway, arterial, local), and an id uniquely identifying the road in an external mapping system. They want to know everywhere a road crosses a line (i.e. where the traffic must wait for a train, or a train for the traffic, not where there are bridges or tunnels separating the two), and for each crossing, the type of crossing (e.g. uncontrolled, traffic-controlled) and, if it is adjacent to a station, to which station it is adjacent. Only one crossing between a particular line and a particular road should be allowed.

Question 5. [30 marks]

This question concerns a vending machine, represented by the UML class below and described in the textbox below.

class Vending_Machine

Vending_Machine
-    state :{waitingForSelection,waitingForCoins,refundingCoins,dispensingProduct,outOfOrder} -   selected :Product -   runningTotal :int
+   select(Product) :void -   available(Product) :boolean -   price(Product) :int -   dispense(Product) :void +   dispensingComplete() :void +   insert(Coin) :void -   value(Coin) :int +   cancel() :void -   refundCoins() :void +   refundingComplete() :void -   error() :void +   reset() :void

When the vending machine is turned on, it will wait for the customer to select a product. It will then wait for the customer to enter sufficient coins to pay for the product. Once the total value of the coins is greater or equal to the price of the product, the vending machine will dispense the product (no change is issued); prior to this, the customer can change their mind and cancel the purchase which will refund any coins inserted and ready the machine for another purchase. At any stage, an error may occur putting the machine into an error state until the problem is corrected and an internal reset button is pressed by a technician.

Question 5 Part 1. [20 marks]

Draw a state machine for this vending machine, annotated to show use of the methods and attributes shown in the corresponding class below.
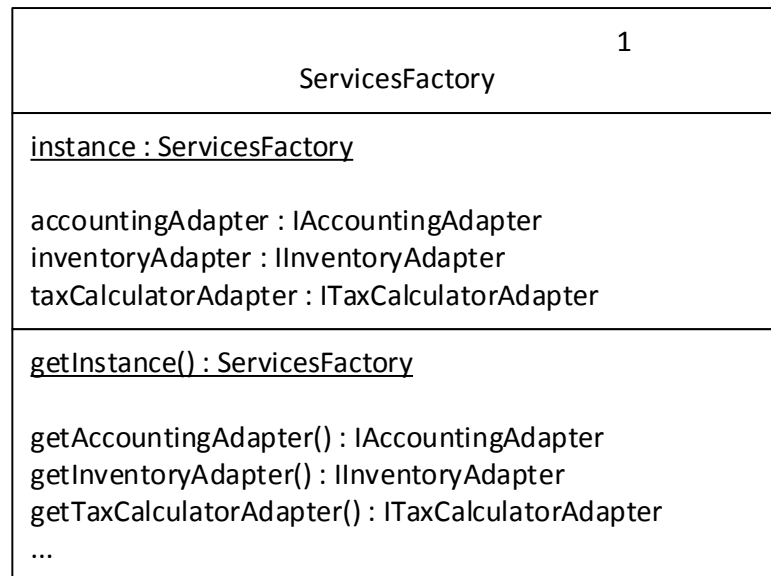
Question 5 Part 2. [10 marks]

Draw a sequence diagram covering a scenario where a customer encounters an error after selecting an apple juice (price $2:80), but successfully completes the purchase using a $1 coin and $2 coin after a technician fixes the problem. Include all method calls but lower level detail (e.g. attribute assignment) is not required.

Question 6. [6 marks]

Two GRASP patterns/principles try to trade-off between representing an object-oriented system as a single complex object and representing it as many simpler but highly connected objects. Name and describe these patterns and the trade-off they represent.

Question 7. [18 marks]

Below is an incomplete UML class. The questions below relate to this class.

```
                                        1
                  ServicesFactory
─────────────────────────────────────────────
 instance : ServicesFactory

 accountingAdapter : IAccountingAdapter
 inventoryAdapter : IInventoryAdapter
 taxCalculatorAdapter : ITaxCalculatorAdapter
─────────────────────────────────────────────
 getInstance() : ServicesFactory

 getAccountingAdapter() : IAccountingAdapter
 getInventoryAdapter() : IInventoryAdapter
 getTaxCalculatorAdapter() : ITaxCalculatorAdapter
 ...
```

Question 7 Part 1. [12 marks]

This class represents the application (primarily) of three patterns. Name these patterns and briefly describe for each: the problem that they address, and how they address that problem.
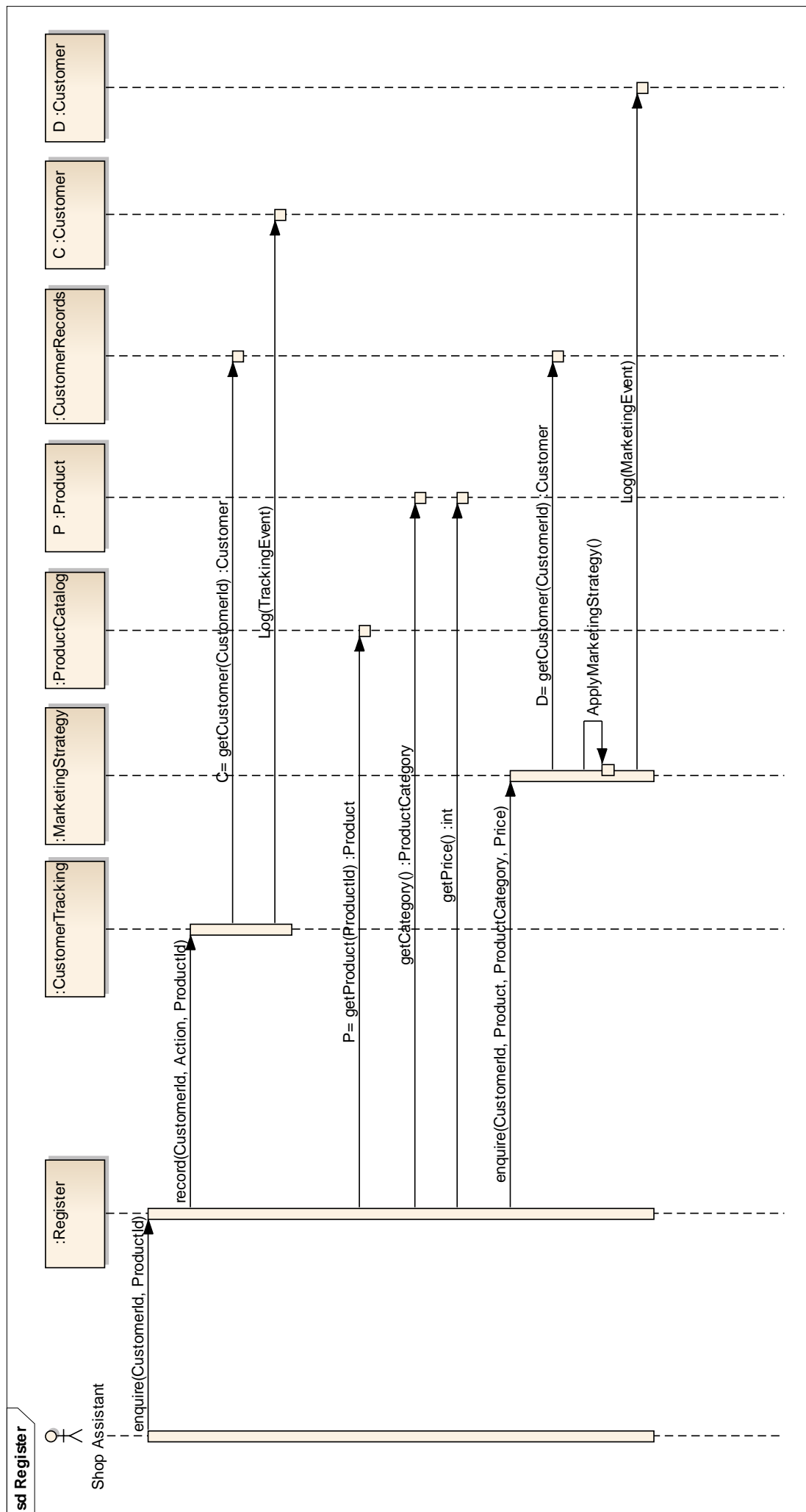
Question 7 Part 2. [6 marks]

Write the method *getAccountingAdaptor()* in Java, assuming a data driven design which supports configuration of the software on a per site basis. You may assume the availability of reasonable library routines to support your implementation.

Question 8. [18 marks]

Consider the sequence diagram below, relating to a store described in this textbox:

> A particular store supports customers enquiring about products: the store responds to this enquiries later. The store applies a marketing strategy to decide how to respond to these enquiries, and logs the resulting marketing event. As the store recognizes the importance of its customers, it also logs every interaction with a customer, including the product enquiry itself. The store wants customers to be able to make these enquiries via their web portal, and would like to take the opportunity while making changes to clean up their system design.

Draw a revised sequence diagram for an in-store product enquiry, and a sequence diagram for a product enquiry via a web portal. You may use UML frames to support this task. The functionality of the in-store enquiry should not change.
Briefly explain the principles behind the system changes you have made.

sd Register

Shop Assistant

:Register

:CustomerTracking

:MarketingStrategy

:ProductCatalog

P :Product

:CustomerRecords

C :Customer

D :Customer

enquire(CustomerId, ProductId)

record(CustomerId, Action, ProductId)

C= getCustomer(CustomerId) :Customer

Log(TrackingEvent)

P= getProduct(ProductId) :Product

getCategory() :ProductCategory

getPrice() :int

enquire(CustomerId, Product, ProductCategory, Price)

D= getCustomer(CustomerId) :Customer

ApplyMarketingStrategy()

Log(MarketingEvent)

End of the Exam Questions

End of the Exam Paper