

Ludwik Kuzniarz
Blekinge Institute of Technology
School of Computing
Karlskrona

August the 21st, 2015

Course PA1415
Object Oriented Design

Points

Q 1: 25	Q 2.1 : 8	Q 2.2 : 7	Q 3.1 : 6	Q 3.2 : 24	Total

Grade

BTH	ECTS

----- Explanations -----

Questions.

For the multiple choice questions your task is to indicate the following statements as *true* T or *false* F by placing the appropriate letter indicator in the [].

For instance

[T] John likes Mary

indicates that the statement is true, or more precisely you think it is true.

John is

[F] Swedish

[T] English

[F] 5 years old

indicates the John is not Swedish, he is English and he is not 5 years old,

If you know that John is German and 20 years old, you should made the following indications:

John is

[F] Swedish

[F] English

[F] 5 years old

For the problem questions your answers should be written in the predefined marked places

either labelled boxes

or along labelled lines

Well structured answers will be appreciated.

Marking

Every question, just after the question number, has a number of points allocated for that question.

If all entries for the question are marked correctly you obtain that number of points. For any wrong answer for the question one point is subtracted from the number of allocated points but no negative points are generated. It means that if a question has 2 points allocated and has three places to mark T or F then when you make one error you get 1 point for that question, when you make two errors you get 0 points and when all the answers are wrong you also get 0 points.

Test is worth 70 points, 36 point is passed, 55– very good.

Allowed books

English – Swedish dictionary

A remark on drawings

In the case of tasks that require producing drawings – conceptual models, state diagram, class diagram – please draw first your draft solutions on a spare paper and then redraw them on the marked area on the examination paper trying to arrange the elements (and especially connecting lines) of the picture so that the models were easy readable.

So, good luck!

1. Knowledge**27 p**

1. 2 p
The tasks performed during the Requirements include
- identification of processes of using the system
 - drawing UseCase Diagrams
 - writing operation contracts for the system operations
 - defining the structure of the software system
2. Class 2 p
- may represent a concept in a domain
 - may represent software element
 - must have attributes
 - must have operations
3. Conceptual Model shows 2 p
- attributes of the concepts,
 - operations performed by the concepts,
 - relationships between concepts,
 - exchange of operations between concepts.
4. 2 p
Sequence Diagram:
- can be used to describe the activities performed by an actor within the use case
 - can be used to describe how an object changes states
 - shows objects, links between objects and messages sent between objects
 - can be replaced by a collaboration diagram
 -
5. 2 p
State Diagram
- contains states, events and transitions
 - may be structured
 - may illustrate how attributes are changed in response to events
 - may describe conditions for changing states
6. Cohesion and coupling 2
- Coupling is the measure of how much focused are the operations in a class
 - Cohesion is the measure of how much dependent is one class on other classes
 - Cohesion should be kept high
 - Coupling should be kept low

7.

6 p

Lars, who is a person, has two cars : Ford Focus and Opel Vectra.

Magnus has no car.

Ford Focus belongs only to Lars but Opel Vectra belongs also to Mary.

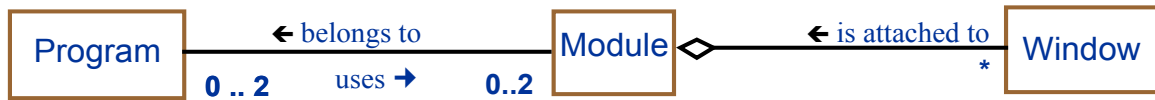
This specific situation in a domain conforms to (is allowed by) the following conceptual model:

- []
- []
- []
- []
- []
- []
- []
- []
- []
- []

8.

7 p

Consider a domain described by the model



Is the situation described below consistent with (allowed by) the above model:

- Orphan:Module does not belong to any Program,
- There is only one Single:Module belonging to Test :Program,
- M :Module belongs to Test1:Program and Test2:Program,
- Game :Program uses two modules GUI:Module and Controller :Module,
- Lonely:Window is not attached to any Module,
- A Common: Window can be removed from M1:Module and attached to M2:Module,
- A :Module belonging to Strange:Program has no Windows attached to it.

Are the following statements true or false?

- There must be at least one Module belonging to every Program,
- The same Module cannot belong to two different Programs,
- The number of existing Modules and the number of existing Programs must be the same,
- Every Window must be attached to a Module,
- The number of Windows cannot be smaller than the number of Modules.

2. Design patterns**16 p****2.1. Theory****8 p**

1.

3 p

Patterns, Frameworks and Idioms

- patterns are more general and abstract than frameworks,
- patterns are more primitive than frameworks,
- framework can employ several patterns,
- frameworks are partially completed software systems,
- idioms are related to a specific programming language.

2. Observer Pattern suggests a solution for

2 p

- separating user interface and internal information representation
- managing several views of the same object
- keeping cohesion low
- keeping cohesion high

3.

3 p

Model View Controller

- is an architectural pattern,
- is used to properly structure Use Cases,
- is a pattern used in Conceptual Modelling,
- is a pattern used in the Development Phase to model the overall structure of the system,
- must be used together with the Observer Pattern.

2.2. Practice**7 p**

Consider an Object can store an integer number, and which potentially can read or write, but it can perform only one of the two potential actions at a specific moment. He can be either in the **state** of reading and then the action it can perform is **read** the number x or in the state of writing and then the only action it can perform is **write** the stored number.

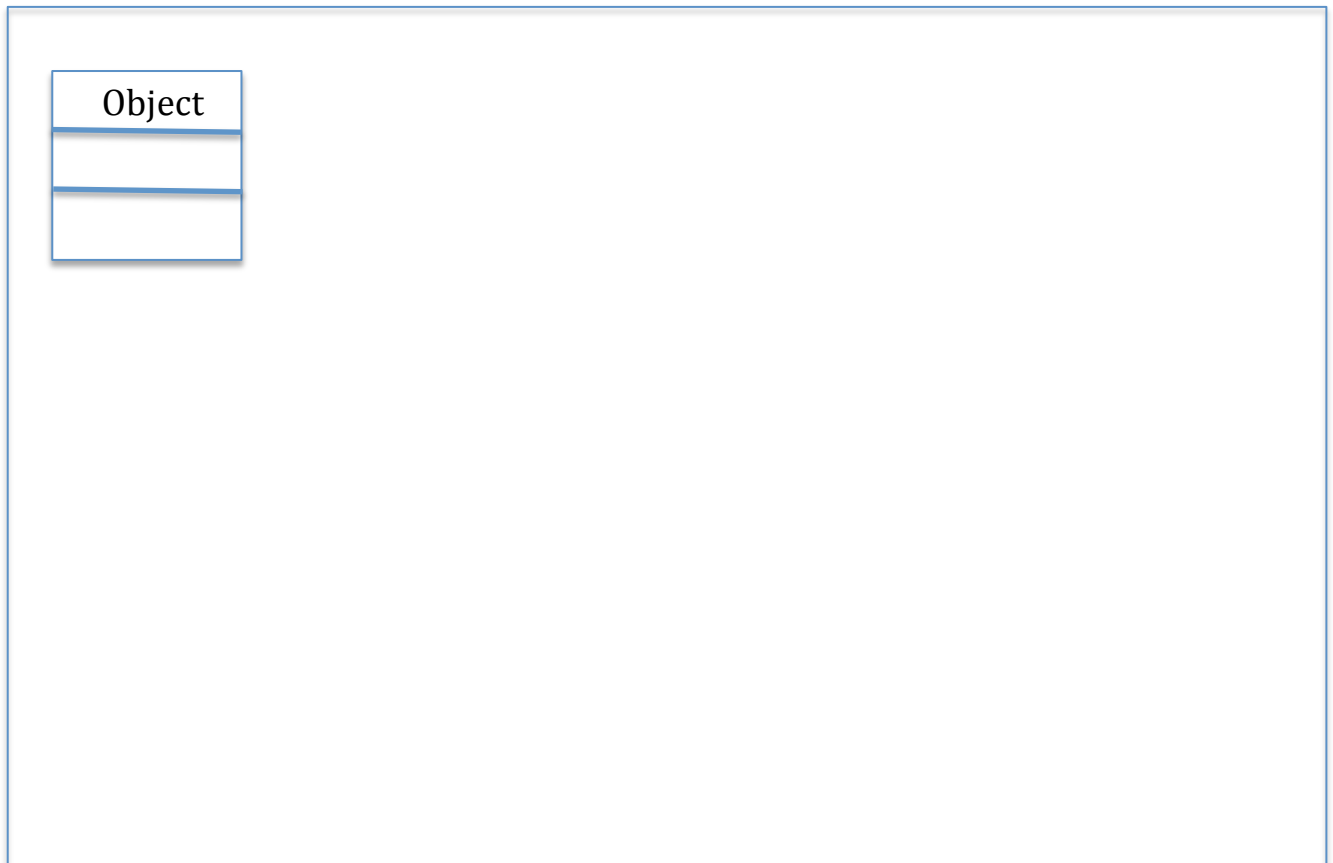
A justified way to cope with this problem is to use a proper design pattern.

a. which design pattern should be used to solve the problem? (just give the name of the pattern)

.....

1 p

b. Complete the class diagram bellow in such a way that it presents a proper usage of the proper pattern – include appropriate classes with appropriate attributes and method, and also appropriate relations between the classes (associations, inheritance, ..)

6 p

3. Skills**30 p****3.1. Modelling Behaviour****6 p**

Car transmission system can be in one of three basic states:

- standing – Stop gear,
- going backward – Reverse gear and
- going forward – Forward gear.

Driver (user of the system) can push the following buttons:

- R – reverse button,
- F – forward button,
- S – stop button

(and in consequence generate an appropriate event).

When system is in the state of going forward it can be going on the First, Second or Third gear.

When the car is going forward the driver can press **Up** or **Down** button.

Pressing **Up** button changes the gear one level up – from 1 to 2, from 2 to 3,

pressing the **Down** button changes gear from one level down : from 3 to 2, from 2 to 1.

From Stop driver can go to both states – to going Forward and to going Backward - by pressing the appropriate button.

When going the car can at any moment go to the Stop – by pressing S.

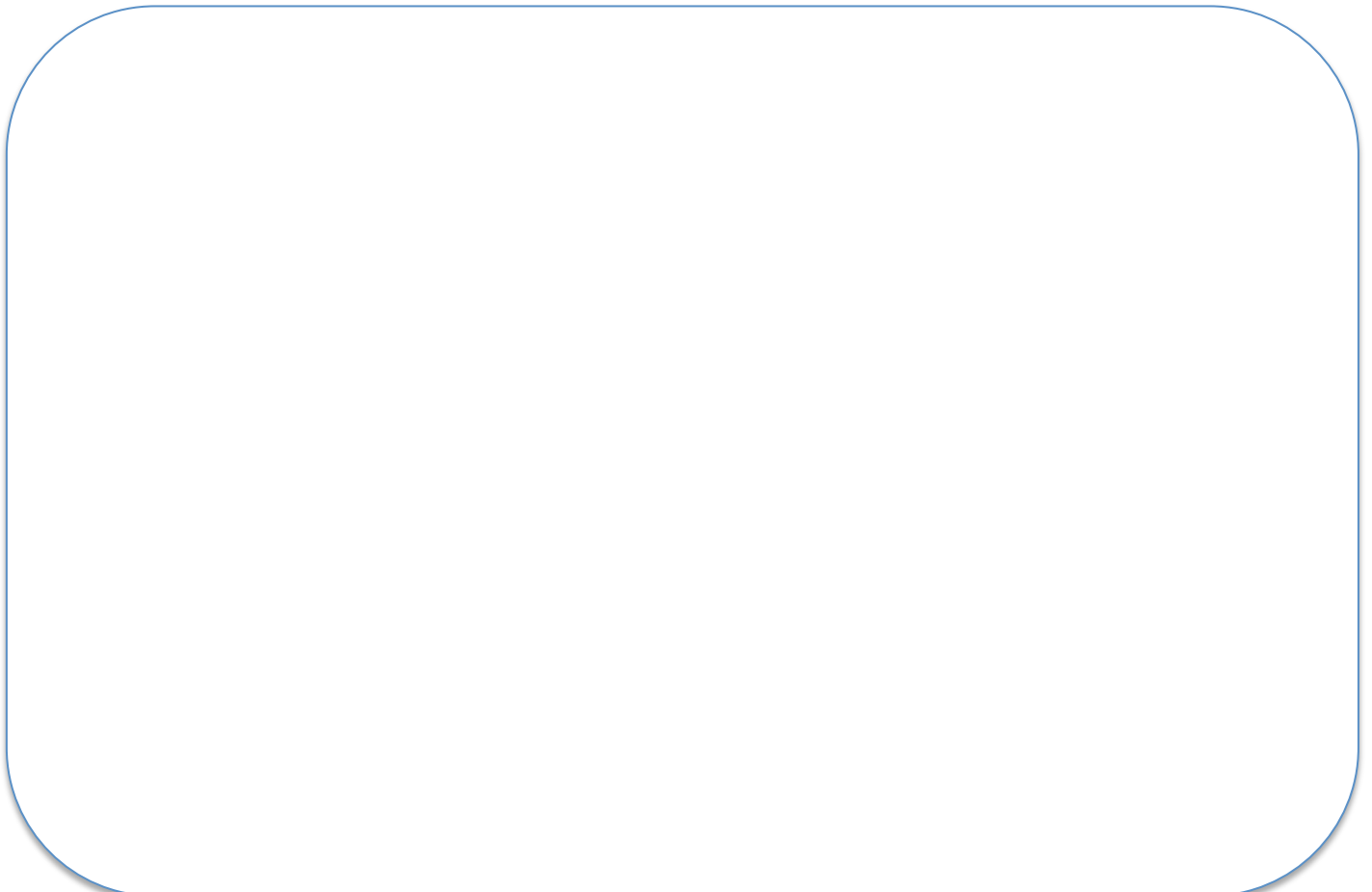
It is not possible to go from going forward to going backward. This can be only possible by first going to Stop and then from Stop to Backward.

Similar applies to going from Backward to Forward – it is not possible to go directly from Backward to Forward – first one has to change to Stop (press stop button) and then change to Forward (press forward button).

Changing from Stop to Forward sets the transmission system automatically in gear one.

Draw a State Diagram modelling the behaviour of the transmission system.

Structured solutions will be rewarded.



Object-Oriented Design**24 P**

From the DrinkMaker you can obtain hot *drinks, coffee* and *tea*.

You can pay only with *coins*. The DrinkMaker accepts the coins (1, 5 and 10 SEK).

In the machine there is a hot *water tank* and *two containers*, one for the tea and one for the coffee. There is a special *heater* in the tank to keep the water at the boiling temperature.

The process of buying a drink consists of two steps – first you compose your drink, then you pay for the drink. After successful payment the drink is prepared and delivered.

To buy a drink you have start the machine.

Then the available drinks are displayed, together with their processes.

You pick the drink and the machine displays the available additional ingredients that can be added to the chosen drink – sugar and milk – together with their process.

You may pick the ingredients and then you proceed to the payment.

During the *payment* your insert coins into the coins slot until the enough total is inserted.

The machine identifies each coin and displays the amount of money left (to be paid). When the total amount of the inserted coins is grater than the price of the drink, the machine gives change.

The DrinkMaker machine is always able to return even change (simplification).

After successful payment, the machine composes the drink, delivers the drink to the client and goes the waiting state.

At any time you can cancel the transaction and obtain the money back.

The drink is made by putting the tea or coffee together with the ordered ingredients in the plastic cup and filling it with hot water.

The DrinkMaker is maintained by a *service* man, who periodically visits the machine in order to perform the *maintenance* - he *refills* the ingredients for preparing drinks, *collects* money and can *change* the prices of drinks.

To do the maintenance, the service man opens the machine with a special key, which sets it to the service mode. After the maintenance the service man closes the machine and exits from the service mode.

(Bonus: You may also consider that the DrinkMaker is able to send a pager message to the service man informing about emergency situation such as: lack of ingredients, overflow of cash, and insufficient cash for change).

3.2. Your tasks**24 p**

- | | |
|---|-----|
| A. Identify Actors | 1 p |
| B. Identify Use Cases | 2 p |
| C. Draw Use Case Diagram | 2 p |
| D. Write extended version of Buy_Drink Use Case | 4 p |
| E. Draw Conceptual Model | 6 p |
| F. Draw System Sequence Diagram for Buy_Drink Use Case | 3 p |
| G. Identify System Operations based on Buy_Drink Use Case | 2 p |
| H. Write a Signature for the Insert_Coin System Operation from Buy_Drink use case | 1 p |
| I. Write a Contract for the Insert_Coin System Operation (both Text and Stage and Curtin) | 3 p |

A. Actors

1 p

.....

.....

.....

.....

B. Use cases

2 p

.....

.....

.....

.....

.....

.....

.....

.....

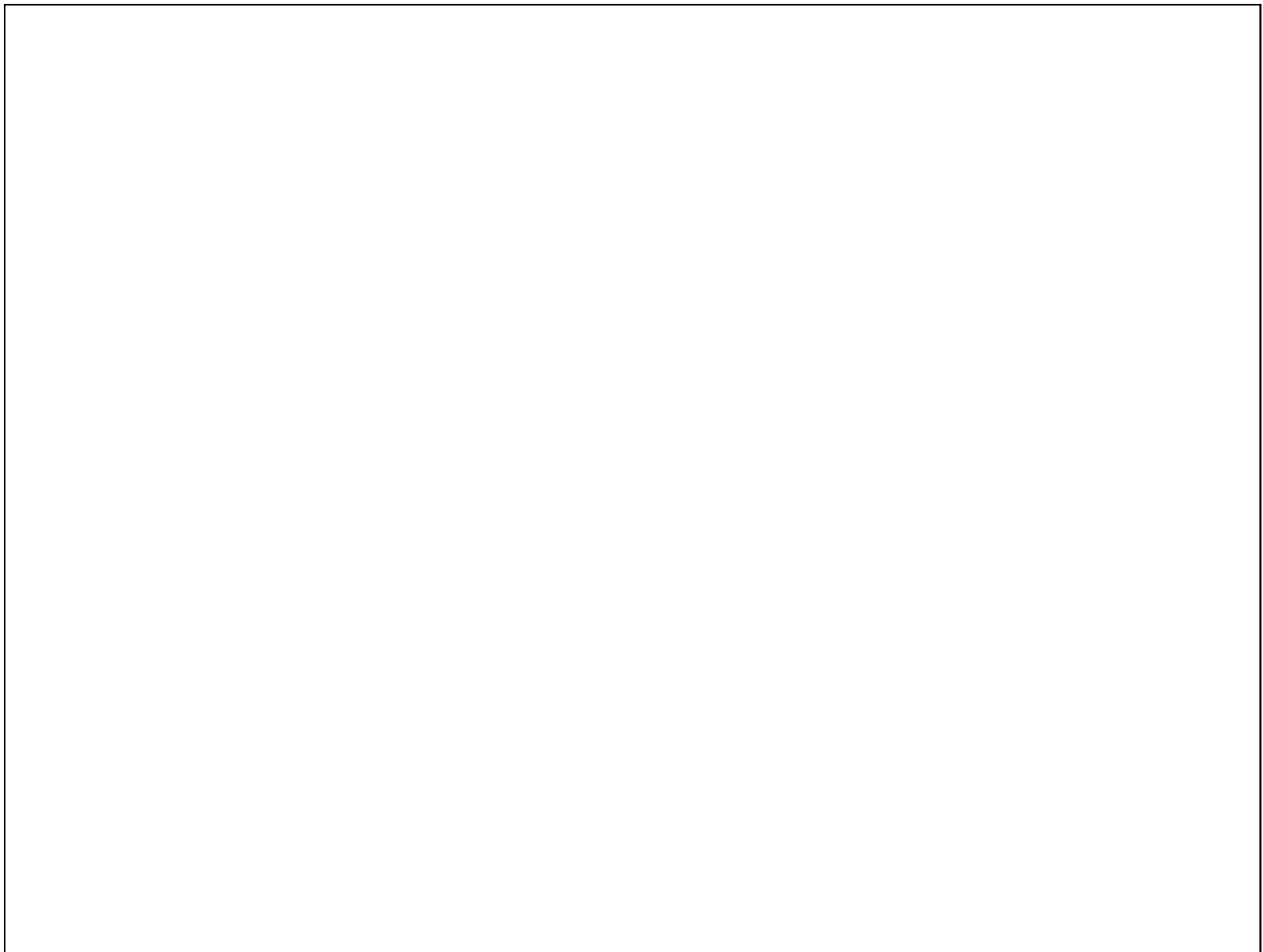
.....

.....

C. Use Case Diagram

2 (+1) p

(Consider Structuring Use Cases – big process with subprocesses)



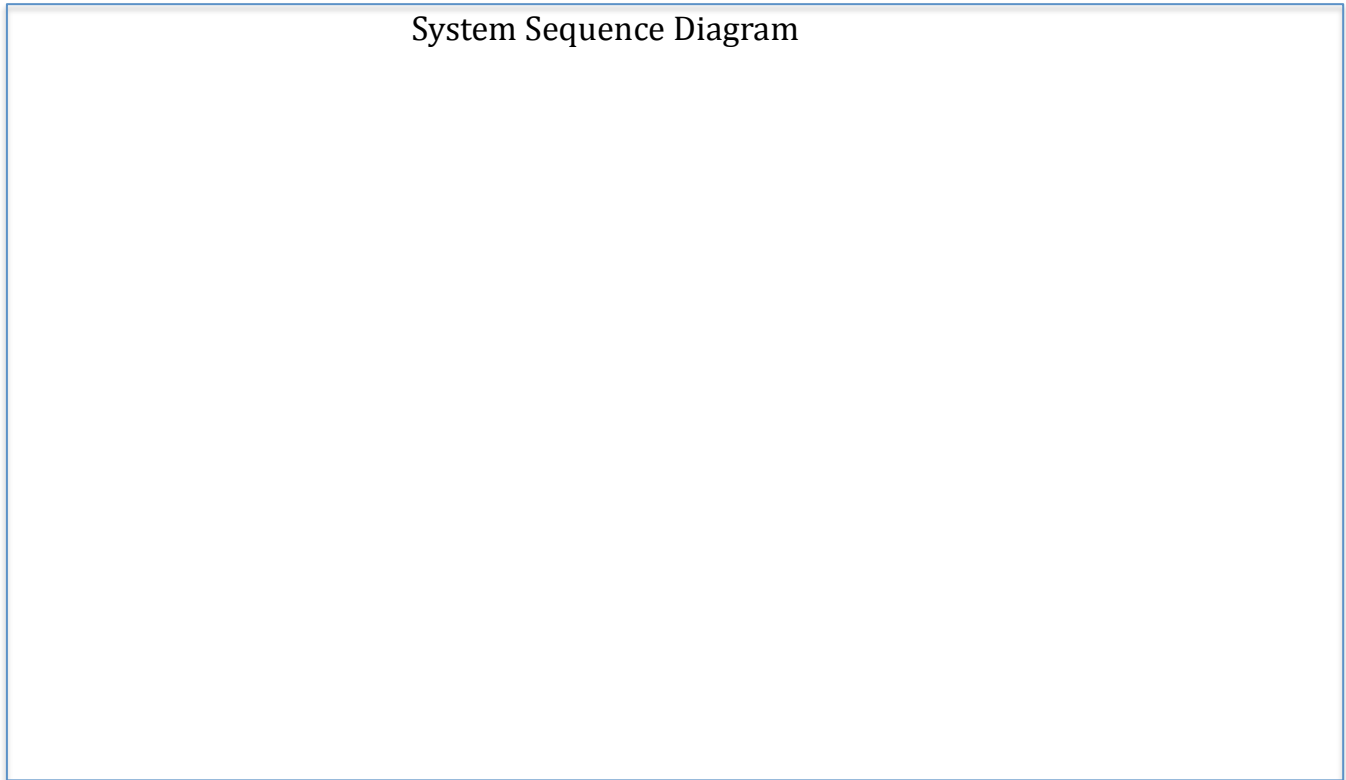
E. Conceptual Model = Domain Model 6 p

(Use inheritance, different types of aggregation; model things as objects rather than attributes)

Domain Model

F. System Sequence Diagram for **Buy_Drink** Use Case

3 p



G. System operations based on **Buy_Drink** Use Case

2 p

.....

.....

.....

.....

.....

.....

.....

.....

H. Signature for the **Insert_Coin** system operation

1 p

The operation is supposed to verify if the coin is a proper coin and to update appropriate container
(Signature for an operation specifies the name of the operation, arguments together with their types and returned value)

.....

.....

- I. Contract for the **Insert_Coin** operation
(both Text and also Stage and Curtin)

3 p

*The operation is supposed to verify if the coin is a proper coin and to update appropriate container
(and may to handle also some other situations if you think/consider so)*

Text

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Stage-Curtin

.....

Exam spare sheet – for your drafts

Exam spare sheet – for your drafts

Exam spare sheet – for your drafts