Student Name:

Student Number:

Faculty of Science
Final Examination

Computer Science 308-304B
Object-oriented Software Design

Examiner: Prof. Hans Vangheluwe — Friday, April 19^{th}, 2002

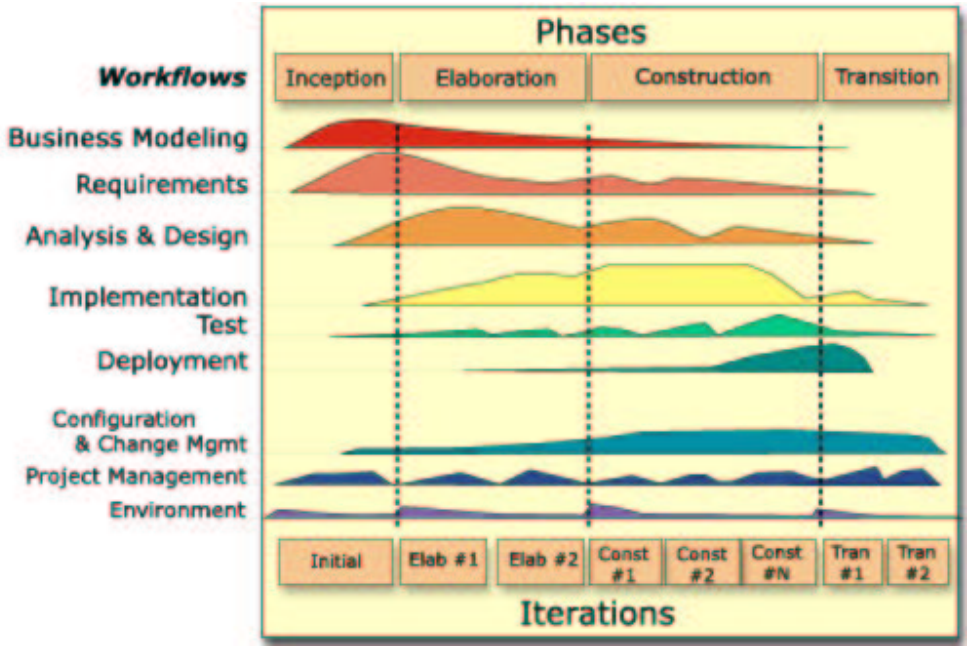
Associate Examiner: Prof. Karel Driesen — 9:00 – 12:00

INSTRUCTIONS:

1. Answer all questions directly on the examination paper.
2. No notes, books, calculators, computers or other aids of any type are permitted.
3. Translation dictionaries may be used.
4. The exam has 21 questions on 15 pages.
5. Attempt all questions: partial marks are given for incomplete but correct answers.
6. Numbers between brackets [] denote the weight of each question. The exam is out of a total of 60 points.
7. Use the back of the last page as scrap (it will be ignored during grading). The rear of the other pages may be used as extra space to answer questions.

Good luck !

(1) [2]



The above figure depicts a software process. Which characteristics do you observe in this process ?

(2) [3]

- What are the characteristics of good unit tests ? Note: the answer is *not* the different types of tests.

- Suppose we want to test the class `Square` which encapsulates a square with side length `side`. The class has the following public methods with the obvious meanings:
  - `setSide(Real)`
  - `setSurface(Real)`
  - `getSide() -> Real`
  - `getSurface() -> Real`

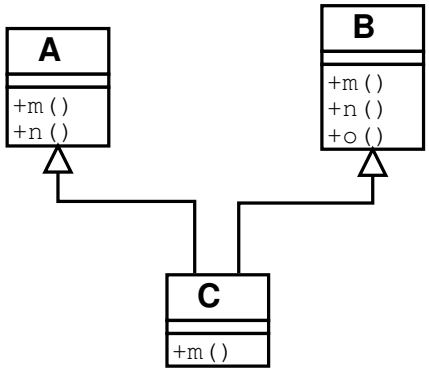
  For the above, give a small example of each different type of test which needs to be done.

(3) [1]

Name the 9 generally accepted features of Object-Oriented systems (no explanation needed).

1.
2.
3.
4.
5.
6.
7.
8.
9.

(4) [3]



1. Referring to the above figure, explain multiple inheritance and the problem it introduces.

2. How is the problem resolved in an OO language such as Python ?

3. What does one call the relationship between method `m()` in classes `A` and `C` ?

(5) [4]

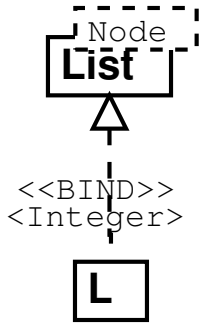
- Draw in UML notation, examples of *inheritance*, *aggregation* and *composition*.

- How would you check, during design, whether a relationship is an inheritance relationship or not ?

(6) [3]

Explain and compare *polymorphism* and *overloading* of class methods. Illustrate by means of a UML diagram.

(7) [1]



Which feature of some OO systems is shown above ? Explain its meaning briefly.

(8) [1]

- What is the difference between class variables and instance variables ?

- What is the UML notation for both (give an example) ?

(9) [8]

Draw the *class diagram* for Statecharts (syntax).
Unless explicitly mentioned otherwise, all class properties have public visibility. A `Statechart` is always composed of *exactly one* (`toplevel`) `OrthogonalComponent`. A `Statechart` can refer to its `toplevel` `OrthogonalComponent` but not the other way around. A `Statechart` has private attributes `creator:String` and `createdDate:Date`. Furthermore, it has a public `draw()` method. Any `OrthogonalComponent` has a `name:String` which must be *unique* at each level (inside a `CompositeState`). An `OrthogonalComponent` contains 0 to many `GeneralisedState` objects (the number may change as a function of time). It is possible to navigate from `OrthogonalComponent` to `GeneralisedState` and vice versa. `GeneralisedState` can be classified as either `SimpleState`, `PseudoState`, or `CompositeState`. These are the *only* three possibilities and a `GeneralisedState` object is always an instance of *exactly one* of these three. A `SimpleState` object has a

`type` attribute which indicates whether the state is NORMAL, INITIAL, or TERMINAL. A `PseudoState` is either a `ConditionState` or a `SelectState`. A `CompositeState` contains 1 or more `OrthogonalComponents`. Navigation in both directions is possible. `GeneralizedState` (and any subclass) objects are connected by means of `Transition` objects. Transitions are *hyper-edges* in a hypergraph with `GeneralizedState` objects as nodes. This fact determines the multiplicities ! A `Transition` has three `String` attributes: `event`, `guard`, and `action`. `GeneralizedState` has a `draw()` method which is however implemented in its sub-classes.

(10) [1]

Explain “cascading delete” of an object.

(11) [2]

Draw a collaboration diagram for two `Client` objects `client1` (first) and `client2` (later) registering themselves with an appropriate message with a `Server` object `server`. Some time later, the `server` will *callback* both clients and invoke their `do_it()` methods. While doing so, the `server` will make it possible for the client to know *which object* called its `do_it()` method.

(12) [2]

Draw a State Automaton which recognizes the regular expression `^(AB)*C+F$`

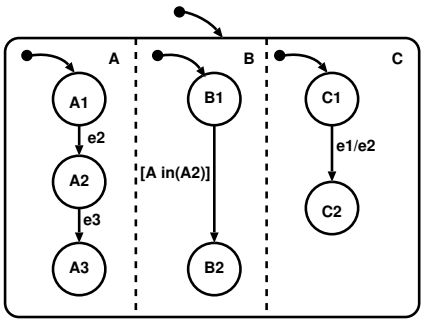
(13) [2]

Draw a Deployment Diagram for an `AccountingComponent` with interfaces `UserServices` and `ManagerServices` implemented on a `WindowsNTServer`, a `UserApps` component accessing `AccountingComponent`'s `UserServices`, running an a `PCWindows2000` machine. Communication takes place over a 100Mbps TCP/IP LAN.

(14) [7]

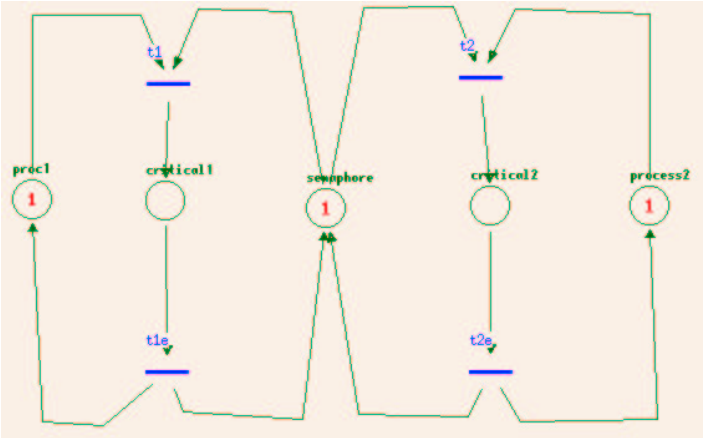
1. Briefly describe the Statechart formalism. In particular, what do Statecharts add to State Automata ? Draw a simple example for each of the "features".

2. Write the state trajectory for the Statechart below receiving the input event segment $e_2 e_2$ as input.



(15) [2]

- Draw the state automaton corresponding to all possible behaviours of the Petri net below



- Explain briefly how one can prove mutual exclusion of the two critical sections.

(16) [2]

- In good OO design, what is the relationship between the state-space of a class C and the state-space of a subclass S of C ?

- Illustrate by means of an example.

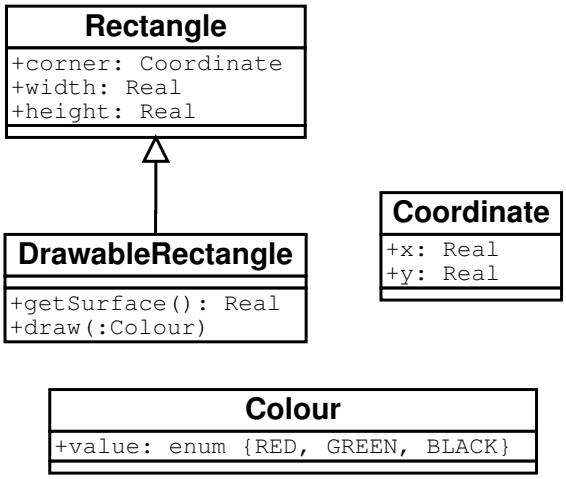
(17) [3]

Explain covariance and contravariance by means of an example.

(18) [3]

1. Explain encumbrance in your own words.

2. Give the indirect class-reference set of `DrawableRectangle` in the design given below.



3. Give the indirect encumbrance for `DrawableRectangle`.

4. What does a class in a high domain (the application domain for example) but with a low indirect encumbrance indicate ?

(19) [5]

1. Draw a Class Diagram for the Observer Pattern. Use concrete `SpreadsheetData` as subject and `GridView` and `PlotView` as observers.

2. Assume one subject `sd:SpreadsheetData` and two observers `gv:GridView` and `pv:PlotView`. `gv` sets data in `sd` *twice* in sequence. Draw *two* sequence diagrams corresponding to the two alternatives of who is responsible for notifying observers when a subject changes.

(20) [2]

Describe how to support Undo and Redo when using the Command Pattern.

(21) [2]

Draw the Class Diagram for the Composite Pattern.