

# EXAM

## TDDC18 Component-based Software

2007-06-04, 08:00–12:00

**Examinator:** Christoph Kessler

**Jour:** Christoph Kessler (070-3666687, 013-282406)

### Hjälpmedel / Admitted material:

Engelsk ordbok / Dictionary from English to your native language

### General instructions

- This exam has 10 assignments and 5 pages, including this one.  
Read all assignments carefully and completely before you begin.
- It is recommended that you use a new sheet of paper for each assignment. Number all your sheets, and mark each sheet on top with your name, personnummer, and the course code (TDDC18).
- You may answer in either English or Swedish.
- Write clearly. Unreadable text will be ignored.
- Be precise in your statements. Unprecise formulations may lead to a reduction of points.
- Motivate clearly all statements and reasoning. Explain calculations and solution procedures.
- The assignments are *not* ordered according to difficulty.
- The exam is designed for 40 points. You may thus plan about 5 minutes per point.
- Grading: U, 3, 4, 5. The preliminary threshold for grade 3 is 20 points.

For exchange students (with a 'P' in the personnummer), we will use the ECTS grading scale.

**OBS Care antagna före 2001:** Om du vill ha ditt betyg i det gamla betygssystemet (U, G, VG) skriv detta tydligt på omslaget av tentan. Annars ska vi använda det nya systemet (U, 3, 4, 5).

## 1. (8 p.) OO technology and design by contract

Consider the following interface definition (pseudocode):

```
interface Catalog {
    function addEntry ( in Type1 item, out Type2 status )
        throws CatalogSizeExceededException;
    function listEntries ( );
}
```

where `Type1` and `Type2` denote types, `in` denotes input parameters and `out` denotes output parameters (results) of a function.

A new class with the following interface `MyCatalog` should serve as a subcontractor of the `Catalog` interface:

```
interface MyCatalog extends Catalog {
    function addEntry ( in Type3 entry, out Type4 st )
        throws ExceptionType5;
    function listEntries ( );
}
```

- (a) Define the terms *covariance* and *contravariance* of parameter types when using a subclass as subcontractor.

Which of these two imposes an extensibility problem in OO-based component-based design, and why?

Demonstrate the covariance and contravariance constraints with the above example code. Specifically, what are the constraints that `Type3` and `Type4` must fulfil in order to guarantee that class `A` fulfils the syntactic part of the contract defined by `Catalog`? (3p)

- (b) Formulate the corresponding syntactic substitutability rule for (checked) *exception types* that can be thrown by a subclass as subcontractor. For instance, consider the exception type `ExceptionType5` in the pseudocode above. What constraints must it fulfill with respect to `CatalogSizeExceededException`? (1.5p)

(Hint: Exceptions are run-time objects (containing information about abnormal termination of e.g. a method call) whose types are usually modeled as classes. Accordingly, there is a tree-like inheritance hierarchy among exception classes along the subtype relation, where a subtype of an exception type represents a more specific exception.

A `catch( XYException e ){...}` clause after a `try` block (that could e.g. contain a call) matches and handles exceptions of type `XYException` or a more specific subtype. In this exercise we only consider checked exceptions that must either be handled by the called method or propagated back to the caller. Exceptions thrown but not handled within a method should be declared and thereby become part of that method's contract (as in the case of `Catalog` above), so that the client of a class/interface statically knows what kind of exceptions may be propagated back from a method call and that it may thus have to handle itself.)

- (c) Components such as language libraries, system functions etc. may be subject to further development e.g. by the component manufacturer even after releases have been shipped to clients who happily use them in their own programs.  
Describe the *Semantic (!) Fragile Base Class Problem*. Give also a simple example demonstrating the problem.  
Which fundamental property of OO languages causes the problem?  
Name a policy for component developers that allows to avoid this problem in practice. (3.5p)

## 2. (3 p.) **Metaprogramming**

- (a) What is *introspection*, and why is it so important for the flexibility of component-based systems? (1p)
- (b) Consider the following C++ program:

```
#include <stdio.h>

template<int n>
struct Factorial
{ enum { RET = Factorial<n-1> :: RET * n };
};

template<>
struct Factorial<0>
{ enum { RET = 1 };
};

void main( void )
{
    printf("Factorial<-1>=%d\n", Factorial<-1>::RET );
}
```

If this program compiles properly, what will be its output if run?

If it does not compile, why not?

(2 p)

## 3. (8 p.) **CORBA**

- (a) Without middleware support, a program written in programming language *A* usually cannot directly call a method written in a different programming language *B* even if they would execute on the same computer. Give one of the technical reasons. (0.5p)
- (b) How does CORBA achieve programming language transparency for static calls? (Describe the principle and those main parts involved in CORBA static calls that are primarily relevant for enabling language transparency. In particular, what does each of these parts do at a call? and which parts are generated, and from what source?) (3p)

- (c) What is an interoperable object reference (IOR) in CORBA? What is its purpose? What data does it contain? Where are IORs created? Name at least one possibility of how a client can get hold of an IOR. (2.5p)
- (d) Writing IDL interface specifications for a large Java application by hand can be quite tedious. How could this task be automatized? Which feature of Java would you need to exploit for this purpose? Sketch *one* appropriate way (general approach and necessary tooling) to realize a program that generates IDL files from the application source code. (2p)

#### 4. (1 p.) **JavaBeans**

- (a) If there was no reflection mechanism in Java, the JavaBeans component model would not work. Why? (1p)

#### 5. (5 p.) **Enterprise JavaBeans (EJB)**

- (a) Enterprise JavaBeans (EJB) allows for separation of business logic from some middleware services. Give two examples of such middleware services and describe how Enterprise JavaBeans handles them. (2p)
- (b) For each Enterprise JavaBean (EJB), the developer must specify which kind of bean it is.
  - (A) What are the 3 different kinds of Enterprise JavaBeans?
  - (B) Why does the EJB system need to distinguish between different kinds of beans? Give 2 examples of what the information can be used for? (3p)

#### 6. (3 p.) **Aspect-Oriented Programming (AOP)**

- (a) The general concept of an *aspect* is useful even when not using AOP.
  - (A) What is an aspect?
  - (B) Give 2 examples of aspects. (2p)
- (b) AspectJ is a powerful tool that aims to solve the problem of crosscutting concerns in Java. What are the greatest new problems that arise with using AspectJ aspects? (1p)

#### 7. (4 p.) **Software Architecture Systems**

- (a) The software system XYZ was implemented using a software architecture system with an ADL (architecture description language) such as ACME or UNICON. Sketch an approach for a tool that automatically checks whether the architecture of XYZ is compliant with the *layered architectural style*. (4p)  
 (Hint: State the condition(s) that must be fulfilled by the connections in a layered architectural style. Then explain how the ADL specification of XYZ provides the information required to check for this condition. Finally, give a simple algorithm (principle, no details) that performs the test. — The layered architectural style is also known as *onion architectural style*.)

8. (3 p.) **Model-driven architecture (MDA)**

- (a) Describe the main concepts of Model-Driven Architecture (MDA, as proposed by the OMG). (2p)
- (b) Explain how MDA supports reuse. (1p)

9. (4 p) **Web services**

- (a) What is a *Service-Oriented Architecture (SOA)*? (1p)
- (b) What is the purpose of each of the following protocols and specification languages used in web services:
  - SOAP
  - WSDL
  - UDDI
  - BPEL?What is the equivalent for each of these in CORBA? (3p)

10. (1 p.) **Invasive Software Composition**

- (a) What is the *compositional interface* of a component in invasive software composition? (1p)

Good luck!