

EXAM

TDDC18 Component-based Software

2006-06-01, 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

Approved *lecture note forms* as submitted during the course. These will be distributed by the examiner approximately 2 hours after the exam started.

Distribution will be easier for us if you put your ID card well visible on your desk.

General instructions

- This exam has 11 assignments and 4 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. (5 p.) OO technology

- (a) Class *A*, which features a single, externally visible method *m* with the signature `Type0 m (Type1 param1, Type2 param2)`, has been used in a large legacy software system for years but became too slow for the increased throughput and should be replaced. There are several faster candidates on the component market that promise to be able to replace *A*. Apart from performance aspects, how can we make sure that a class *B* “fits” as a replacement of *A*? Formally speaking, what is the condition for a class *B* to be able to substitute class *A* in all its uses (syntactic substitutability)? (2p)
- (b) Describe the *Syntactic Fragile Base Class Problem* and its implications for the compatibility of binary components compiled from OO programs. (3p)

2. (3 p.) Metaprogramming

- (a) What is *introspection*? Give one example for how introspection is used in a component system. (1p)
- (b) Given a class `DistributedArray` (here is an example of one that stores floats) that should be made generic such that it can be parameterized in the element type.

```
class DistributedArray {
private:
    float localElements[MAX_NUMBER_OF_LOCAL_ELEMENTS];
    unsigned int nrOfElements;
    unsigned int toLocal( unsigned int i ) {...}; //index transformation
public:
    DistributedArray ( unsigned int globalLength ) {...} //constructor
    float getElementAt ( unsigned int globalIndex ) {
        // ... return localElements[toLocal(globalIndex)];
    }
    void setElementAt ( unsigned int globalIndex, float value ) {
        // ... localElements[toLocal(globalIndex)] = value;
    }
}
```

Use *either* C++ templates *or* invasive software composition to provide a generic version (code) of `DistributedArray`, and show (code) how to instantiate from the generic class a `DistributedArray` of long integers. If you do not recall the exact syntax, invent your own and explain the meaning of each construct carefully. (2p)

3. (6 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.) (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)

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

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

5. (4 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. (1p)
- (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.) **COM**

- (a) Give a short description (annotated picture) of the binary representation of a COM component. (1p).
- (b) The COM component model does not support component reuse by inheritance. Instead, reuse is facilitated by *containment* (composition) and *aggregation*. Describe the similarities and differences between containment and aggregation. (2p)

7. (4 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 is the greatest new problem that arises with using AspectJ aspects? (2p)

8. (5 p.) **Software Architecture Systems**

- (a) What is the component model of software architecture systems? (in other words, what are the (major) different building blocks of a software architecture configuration, and what is their specific purpose?) What are the composition technique(s) and the composition language of software architecture systems? (3p)
- (b) Define the term *layered architectural style* (also known as *onion architectural style*) and give an example of a software system with this style. (2p)

9. (3 p.) **Model-driven architecture (MDA)**

- (a) Explain how MDA supports reuse. (1p)
- (b) What are *marks* in MDA? Where do they occur, how are they represented, and what is their purpose? (2p)
Do you see any analogy to certain entities in invasive software composition? (+0.5p)

10. (2 p.) **Web services and parallel computing**

Assume you have got a few dozen Linux PCs with fast interconnect and should use them as a cluster to run a communication-intensive parallel application (e.g., parallel sorting of floatingpoint numbers). Assume that high performance and effective exploitation of the available hardware are very important. Do you consider web services as a suitable middleware for this scenario? If yes, why? If not, why not?

11. (4 p.) **Invasive Software Composition**

- (a) What is the *compositional interface* of a component in invasive software composition? What are the differences between the compositional interface and the *functional interface* of a component? (2p)
- (b) In the COMPOST system, hooks and fragment boxes are strictly typed. What do these types represent, and why is this typing important? (2p)

Good luck!