

EXAM

TDDC18 Component-based Software 2 juni 2005, 08:00–12:00, TER1

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 11 assignments and 3 pages, including this one. Read all assignments carefully and completely before you begin.
- It is recommended that you use a new sheet 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.
- **OBS Care:** Om du vill ha ditt betyg i det gamla betygssystemet (U, G, VG) skriv detta tydligt på omslaget av tentan. Annars kommer vi att använda det nya systemet (U, 3, 4, 5).

1. (6 p.) **Component models**

- (a) An early component system is Unix' *pipes and filters* (also known as shells and pipes or streams and filters). Describe the Unix pipes and filters model by pointing out its component model, composition technique and composition language. (3p)
- (b) What is the main difference between whitebox, blackbox and graybox re-use of software components? Give an example component/composition framework or scenario for each of these. (3p)

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

- (a) What is the difference between static and dynamic metaprogramming? (2p)
- (b) Give one concrete example system for static metaprogramming. (0.5p)
- (c) Give one concrete example system for dynamic metaprogramming. (0.5p)

3. (4 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. (1p)
- (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)

4. (3 p.) **JavaBeans**

- (a) What changes must be made to the class `PieChartDiagram` in order to turn it into a `JavaBean`? (2p)

```
public class PieChartDiagram {  
  
    public PieChartDiagram ( boolean defaultValue ) {  
    }  
  
}
```

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

5. (4 p.) **Enterprise JavaBeans (EJB)**

- (a) Why do Enterprise `JavaBean` developers have to specify if their session beans are stateful or stateless? (2p)
- (b) One could argue about whether Enterprise `JavaBeans` are object oriented or not. What are the arguments against? (2p)

6. (3 p.) **COM**

- (a) How can a client compare the identity of COM component instances? (2p)
- (b) How is instance deletion handled in COM (name the technique, no details)? (1p)

7. (3 p.) **Aspect-Oriented Programming**

- (a) 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)
- (b) What is the purpose of Join Points in aspect oriented programming? (1p)

8. (6 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?) (2p)
- (b) Define the term *blackboard (repository) architectural style* and give an example of a software system with this style. (2p)
- (c) Software architecture systems are said to be a first step towards separation of concerns. Which concerns are separated, and what are the advantages of this? (2p)

9. (2 p.) **Model-driven architecture**

Give a short description of Model-driven architecture (MDA) (main ideas). What is the main motivation for it?

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 float-ingpoint 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 a hook? Explain the difference between implicit and declared hooks. (2p)
- (b) What kind of operation does the method *x* in the following COMPOST program realize? (Give the technical term and a short description.) (2p)

```
public class ClassBox {
    // ...
    public ClassBox x ( Classbox m, String subClassName ) {
        ClassBox s = this.copy ( subClassName ); // copy and rename a ClassBox
        return s.extend(m);
    }
    // where the extend composition operator on ClassBoxes is defined by
    public void extend ( ClassBox e ) {
        this.findHook ("members").extend ( e.findHook ("members") );
    }
}
```