

Data Processing (DP)

Prof. Dr.-Ing. Christian Heller
<christian.heller@ba-leipzig.de>

Contents

Introduction

Style Guide

Logging

File and Directory

Data Structure

Stream

Object Serialisation and Persistence

Object Cloning

Date and Time

XML Processing

Concurrency

Meta Programming and Reflexion

Language Binding

Java Native Interface (JNI)



Contents



- Introduction
- Style Guide
- Logging
- File and Directory
- Data Structure
- Stream
- Object Serialisation and Persistence
- Object Cloning
- Date and Time
- XML Processing
- Concurrency
- Meta Programming and Reflexion
- Language Binding
- Java Native Interface (JNI)



Learning Objectives

... to enable computer science students to:

- work with data on file system level
- apply various kinds of data structures
- format and serialise data
- understand multi-threading
- use meta programming

Target Group

- students of computer science

Pre-Requisite

- basic knowledge in computer programming
- familiarity with Object Oriented Programming (OOP)

Software

- Eclipse Integrated Development Environment (IDE)

Curriculum


- 1st term: 28 hours
- 1st term: 50 hours
- 2nd term: 54 hours
- 3rd term: 48 hours
- 4th term: 42 hours
- 4th term: 35 hours
- 5th term: 5+60 hours
- 5th term: 60 hours
- 5th term: 60 hours
- 6th term: 56 hours
- Web Technologies
- Computer Programming
- Data Processing
- User Interaction
- Software Engineering
- Project Management
- Systems Implementation
- Programming C/C++
- CYBOP
- Server-side Technologies




Schedule

 Revision: on blackboard (15-30 min)

 Lecture: new matter (60-75 min)

 Break (15-30 min)

 Presentation: done by students (2 x 40 min)

 Break (15-30 min)

 Exercise: solving example tasks (90 min)

 Self-Study (unlimited, > 50 % in a study ;-)

Examination

- presentation during semester ≤ 30 min per student
- examination at end of semester 180 min
 - 120 min DP
 - 60 min SQL
- work with files and directories
- collect data in containers
- use byte and character streams, serialisation, cloning
- apply threads and meta programming

Assessment

- 20 % presentation
- 80 % examination (DP + SQL)

Contents



Introduction
Style Guide
Logging
File and Directory
Data Structure
Stream
Object Serialisation and Persistence
Object Cloning
Date and Time
XML Processing
Concurrency
Meta Programming and Reflexion
Language Binding
Java Native Interface (JNI)





Beautifying

```
int foo(int k) { if (k < 0 || k > 2) { printf("out of range\n");  
printf("this function requires a value of 1 or 2\n"); } else {  
printf("Switching\n"); switch (k) { case 1: printf("1\n"); break; case  
2: printf("2\n"); break; } } }
```

```
int  
foo (int k)  
{  
    if (k < 0 || k > 2)  
    {  
        printf ("out of range\n");  
        printf ("this function requires a value of 1 or 2\n");  
    }  
    else  
    {  
        printf ("Switching\n");  
        switch (k)  
        {  
            case 1:  
                printf ("1\n");  
                break;  
            case 2:  
                printf ("2\n");  
                break;  
        }  
    }  
}
```

Style Guide

- Media design
- Building law
- User interface design
- Source code layout



Examples

- <http://styleguide.bundesregierung.de/>
- <http://publications.europa.eu/code/de/de-8000100.htm>
- <http://www.bbctraining.com/pdfs/newsstyleguide.pdf>
- http://www.wikileaks.org/wiki/WikiLeaks:Style_Guide
- [http://www.freebsd.org/cgi/man.cgi?
query=style&sektion=9](http://www.freebsd.org/cgi/man.cgi?query=style&sektion=9)
- [http://www.gnu.org/prep/standards/standards.html#Forma
tting](http://www.gnu.org/prep/standards/standards.html#Formatting)

Formatted Source Code – Advantages

- consistency
- readability
- understanding
- quickness
- correctness
- cost reduction
- cleanness

Java Code Conventions

- File names and organisation
- Indentation
- Comments
- Declarations and Statements
- White Space
- Naming Conventions

**[http://www.oracle.com/technetwork/
java/codeconv-138413.html](http://www.oracle.com/technetwork/java/codeconv-138413.html)**



Summary

- style guide ensures uniform layout
- applied to media, legal issues, user interface, software
- has many advantages, but requires discipline
- beautifier tools help format source code

Contents



Introduction

Style Guide

Logging

File and Directory

Data Structure

Stream

Object Serialisation and Persistence

Object Cloning

Date and Time

XML Processing

Concurrency

Meta Programming and Reflexion

Language Binding

Java Native Interface (JNI)





Logging

```
import java.io.*;
import java.util.logging.*;

public class Test {

    public static void main(String[] args) throws SecurityException, IOException {

        Logger l = Logger.getLogger("Test");
        Handler h = new FileHandler("Test.txt");
        SimpleFormatter f = new SimpleFormatter();

        h.setFormatter(f);
        l.addHandler(h);
        l.setLevel(Level.FINE);

        l.severe("LogLevel SEVERE");
        l.info("LogLevel INFO");

        // The following is not displayed,
        // since the log level was set to FINE above.
        l.finest("LogLevel FINEST");

        // Alternative output.
        l.log(Level.SEVERE, "LogLevel SEVERE");
    }
}
```