Service Oriented Techniques (SOT) Module Description

In this course you will learn two techniques in Java: (a) synchronous technique - RESTful services. (b) asynchronous Java Messaging Service (JMS), for communication between software applications. **Entry requirements** for this module are:

- successful completion of year 1 of software engineering studies, and
- successful completion of year 2 of software engineering studies, and
- successful completion of year 3 of the software engineering studies, and
- successful completion of the Internship.

Table of Contents

1. Weekly planning 1
2. Assignments 2
   2.1. Assessment Criteria 2
3. Course Assessment 4
4. Learning Outcomes 4

1. Weekly planning

week	Lesson/topic	description
1	**Lesson:** REST	• GET operations • Queue & path parameters
2	**Lesson:** REST	• POST, PUT & DELETE operations • Form parameters • Deploying Java REST service on Apache Tomcat
3	**Lesson:** REST	• Creating custom response in the service • Using GRADLE to manage dependencies and build projects • Composite web services
4	**Assignment:** REST	• Work on the REST assignment (part of final SOT grade) • **Deadline to submit is in Canvas (just before the JMS lesson in week 5).**
5	**Lesson:** JMS	• Sending and receiving JMS messages • Working with QUEUES and TOPICS
6	**Lesson:** JMS	• Asynchronous request-reply communication with JMS • Correlation Identifier pattern • Return Address pattern
7	**Assignment:** JMS	• Work on the JMS Request-Reply assignment (part of final SOT grade) • **Deadline to submit is in Canvas (Monday 06:00 am in week 8).**

2. Assignments

A detailed description of the two (REST and JMS) assignments, which must be handed in for the final SOT grade, will be published on Canvas in weeks 4 and 7.

2.1. Assessment Criteria

General rules are:

- Both assignments (REST and JMS) must be handed in via CANVAS (deadlines are strict).
- For each assignment (REST and JMS) students will get a grade.
- Each project (application) in each assignment has to compile (full IDE projects with all necessary source code, resources and libraries must be submitted) and run (work) on the computer of the teacher (on the latest version of IntelliJ, Apache Tomcat, ActiveMQ or any other IDE/tool you use). If one of the projects (applications) in the assignment does not compile or cannot run on the computer of the teacher (e.g., due to syntax errors, missing libraries, runtime errors, etc.) , then your assignment will not be checked, and your grade for that assignment will be 1. The only exception to this rule is the Tomcat location and port (because you do not know Tomcat location and port on the teacher's computer).
- In both assignments you must implement the required case (which will be published on Canvas when the assignment is published). If another case is implemented (and not the described required case) in an assignment, the grade for that assignment will be 1.

For each assignment there are two types of assessment criteria: [A] **code quality** and [B] **code functionality** (i.e., implemented features).

If your code does not comply to all described code quality criteria, your grade for this assignment will be 2 (regardless of the extent to which your code complies to code functionality criteria). If your code complies to all described code quality criteria, then code functionality criteria will be applied to your code to determine the grade for this assignment.

A. Code quality criteria

All submitted code must comply both (1) to SOLID principles of Object-Oriented Programming and (2) to all items in the Clean Code checklist shown in the table below:

Clean code checklist item	category
Use Intention-Revealing Names	Meaningful Names
Pick one word per concept	Meaningful Names
Use Solution/Problem Domain Names	Meaningful Names
Classes should be small!	Classes
Functions should be small!	Functions
Do one Thing	Functions
Don't Repeat Yourself (Avoid Duplication)	Functions

Explain yourself in code	Comments
Make sure the code formatting is applied	Formatting
Use Exceptions rather than Return codes	Exceptions
Don't return Null	Exceptions

B1. Code functionality criteria for assignment REST

Features in assignment "REST"	grade
Service implements all CRUD operations and used all three types of parameters (path, query and form parameters). Service has an advanced (real-life) example.	5
The service is hosted on Apache Tomcat.	5
Make one Java client (1st client) that calls all methods of the service. The client can be console, GUI or web based. The client must be user-friendly: - user sees (some kind of) menu where he/she can choose which operations to perform (CRUD). - user can input the data for the operation he/she chose (CRUD).	5
HTTP messages for all CRUD operations and all three types of parameters (a document showing and shortly explaining HTTP messages exchanged between the service and the client).	5
Service uses your own custom-made class (e.g., Student, Product) as a parameter or return value.	6
Service uses a collection (e.g., List<Student>, List<Product>) as a parameter or return value.	6
All submitted projects/modules are gradle projects.	7
Create your own custom Response with nice error messages.	8
Make 2nd client in other language than Java (e.g., C#, Mobile, etc.). In this case, you will submit two clients: 1st in Java and 2nd in another language (e.g., C#, Mobile, etc.).	9
Think of and implement a composite service example. There is one "composite" service which implements a business process and calls at least two "basic" services. So, you will make at least three services (one "composite" service + two "basic" services).	10

B2. Code functionality criteria for assignment JMS

Features in assignment "JMS"	grade
Both Requester and Replier applications work using *asynchronous* communication.	5
Pattern Correlation Identifier is implemented (Requestor application shows clearly to which request each reply belongs).	5
Requestor Application and Replier applications have a nice real-life example.	6
Pattern Return Address is implemented.	7
Custom objects (e.g., Student, Order, …) are serialized to JSON (you may choose any JSON library you like) and sent via Messages between Requestor and replier applications.	8
It is possible to start multiple Requestor applications (this is related to Return Address) • without changing the code of Replier application and • without shutting down and restarting the Replier application (i.e., while Replier application is running).	9
Apply pattern Datatype Channel by sending/receiving two types or request/reply data pairs.	10

3. Course Assessment

The final grade of SOT course is based on the REST and JMS assignments:

- Both assignments (REST and JMS) must have a sufficient grade (i.e., at least 6) in order to pass SOT with a sufficient grade.
- The final SOT grade ("**int** sot_grade") will be calculated as follows based on the "**int** rest_grade" and "**int** jms_grade" as shown in Java code below:

```
int sot_grade;
if (rest_grade >= 6 && jms_grade >= 6 ){
    float average_grade = (rest_grade + jms_grade) / 2;
    sot_grade = Math.round(average_grade);
} else {
   sot_grade = Math.min(rest_grade, jms_grade);
}
```

4. Learning Outcomes

Week 1 – REST

- Student can make a RESTful service in Java. The service contains at least four GET methods and uses as parameters and return types of only standard data types (integer, string, char and double).
- Student can use query and path parameters in GET operations.
- Student can make a Java application (called the client application) which calls the RESTful service.
- Student can explain the structure of (RESTful) HTTP messages exchanged between the service and the client.

Week 2 – REST

- Student can make PUT, POST and DELETE operations in RESTful service and client.
- Student can deploy a RESTful service on Tomcat.
- The student can make custom error messages in a RESTful service.

Week 3 – REST

- Student can explain what a composite web service is.
- Student can make a composite RESTful service in Java.
- Student can use gradle to build Java projects.

Week 5 – JMS

- Student can explain the difference between synchronous RESTful technique and asynchronous JMS.
- Student can make Java applications which send and receive JMS messages.
- Student can describe the difference between two types of messaging channels: Queues and Topics.
- Student can use both Queues and Topics for exchanging messages.

Week 6 – JMS

- Student can make a request-reply type of communication with JMS between two Java applications.
- Student can apply the Correlation Identifier pattern with JMS in Java applications.
- Student can apply the Return Address pattern with JMS in Java applications.