MSSE672 – Component Based Software

Student: John Michael Kreski

Instructor: Mohammad Abu Matar

Assignment: Week 5 – XML Parsing + Validation

Date: 08/03/2025

File Name: HWExecution.doc

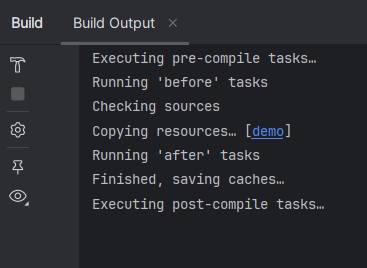
## Summary

This assignment focused on extending the GeometryApp by implementing a standalone ServiceFactory class that reads service class mappings from an XML configuration file (services.xml). The purpose of this activity was to demonstrate XML parsing using the DOM API, validate the XML structure against a custom XSD schema (services.xsd), and dynamically load service implementation classes using Java reflection.

This approach simulates what a .properties-based factory might have done in non-Spring applications, replacing traditional text-based configuration with structured and schema-validated XML. The factory retrieves a class name from XML, verifies it against the schema, and instantiates it only if it implements the QuadService interface. A DummyQuadServiceImpl class was created specifically for this assignment to support simple in-memory behavior outside the context of Spring Boot.

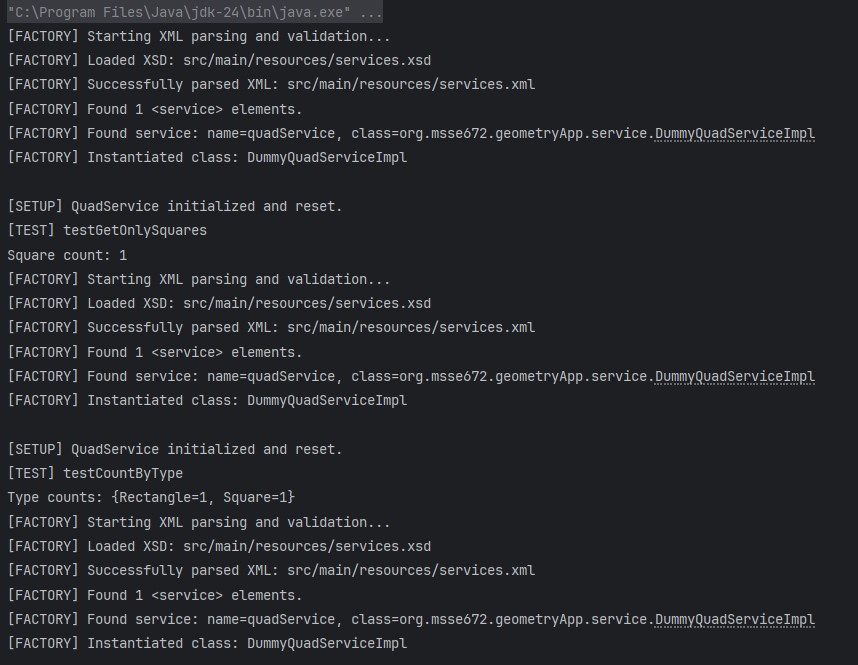
Console logs were used to confirm successful XML validation, parsing, class loading, and instantiation. These outputs were captured using System.out.println() for clarity and testing purposes only, as this feature is not part of the main Spring-driven application runtime.

## Successful Build Compilation

*Screenshot of successful compilation from IntelliJ IDEA.*

## Logs

## Screenshot 1 – Console Output of XML Parsing

*This screenshot displays runtime console output showing print messages for factory and service.*

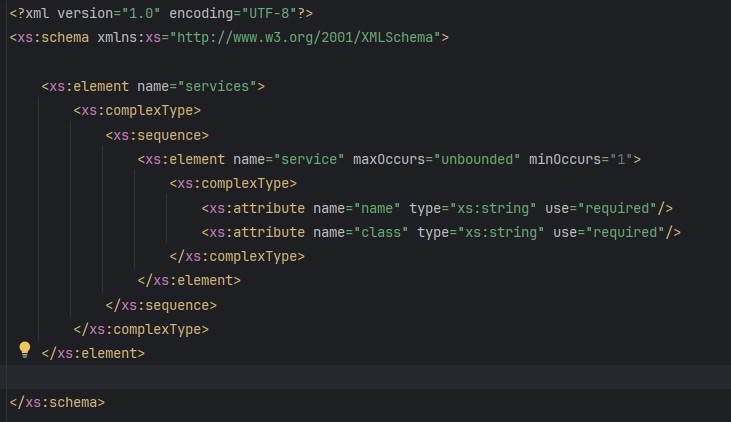
## XML and XSD Files

## Screenshot 1

*Screenshot of services.xml open in IntelliJ showing the <services> and <service> element*

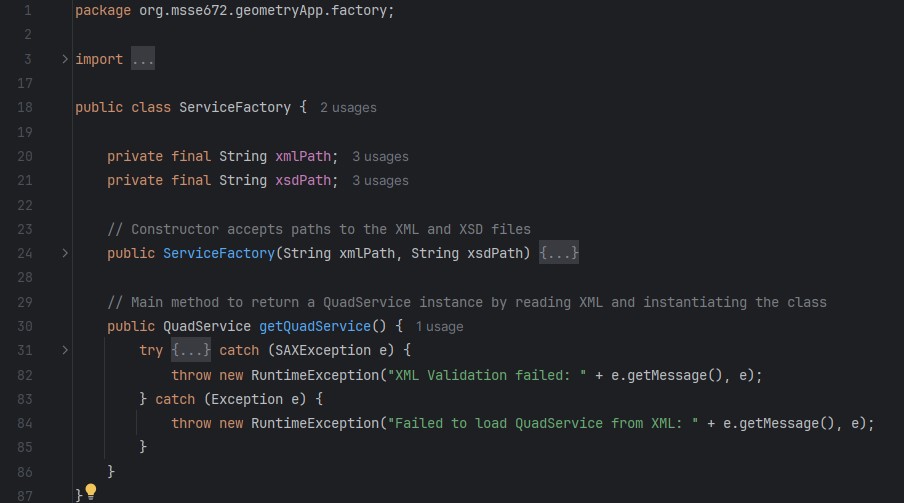
**

## Screenshot 2

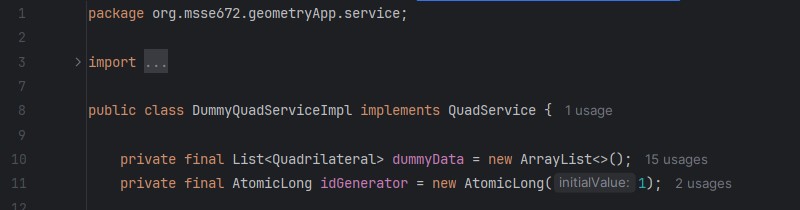
*Screenshot of services.xsd showing the schema definition for validation*

## Source Code View

## Screenshot 1 *Screenshot of ServiceFactory.java showing key logic (XML parsing, validation, class instantiation)*



## Screenshot 2 *Screenshot of DummyQuadServiceImpl.java*



## Refactored Files & Added Files

* **ServiceFactory.java**
  + Standalone Java class that parses and validates services.xml using a DOM parser and an XSD schema. Loads and instantiates service classes via reflection.
* **services.xml**
  + XML file that maps the name "quadService" to a fully qualified class name. Validated using services.xsd.
* **services.xsd**
  + XML Schema Definition (XSD) file that defines the structure and constraints of the services.xml file, ensuring valid formatting before runtime parsing.
* **DummyQuadServiceImpl.java**
  + A simple in-memory implementation of the QuadService interface. Created solely for this assignment to allow standalone testing of XML-driven service instantiation.
* **ServiceFactoryTest.java**
  + JUnit 5 test class that validates the functionality of the ServiceFactory. Confirms the correct class is instantiated and methods like getAllSubmittedQuads() and getLargestSideEverSubmitted() behave as expected.

## Known Limitations & Future Enhancements

* The XML-based ServiceFactory is not integrated into the Spring Boot application and is used for **learning and demonstration** purposes only\*.
* The factory does not support dependency injection or Spring bean lifecycle management and should not replace Spring's native DI mechanisms in production.
* Future enhancements may explore using XML config for external plugin systems or bootstrapping in non-Spring Java applications.

## Notes

* \*Per the professor’s guidance, the XML-based ServiceFactory was implemented as a standalone utility for educational purposes and is not integrated into the Spring Boot runtime. This approach fulfills the assignment’s learning objectives around XML parsing and validation while preserving the integrity of the existing Spring-based architecture.
* The project remains under source control in a public GitHub repository.
* This assignment is delivered as a .zip containing .java source files and a Docs folder per course requirements.
* Console logs and successful JUnit test outputs are included as screenshots in this document.
* Screenshots can be found in:
  + Docs/Week5/assets/