SE3001 – Software Construction & Development

Code is Poetry!

&

Everything boils down to code!

Fall 2023-24

Instructor	Dr. Affan Rauf
Office	HoD SE, Ground Floor, Main Building
Office Hours	TBD
Email	affan.rauf@nu.edu.pk
Telephone	ext. 127
	www.linkedin.com/in/affanrauf

Course Information			
Credit Hours	3		
Lecture(s) per week	2	**Duration**	85 minutes each
Core/Elective	BS SE Core	**Grading Scheme**	Absolute

Course Description

This course assumes that you have already developed object-oriented programming, analysis and design skills. We shall develop working software in Java during this course. You'll learn version control using Git, event-driven programming, writing unit tests in JUnit5, using an issue-tracking system etc. Towards the end of the semester, you shall learn the concepts of code smells, refactoring and selected GoF design patterns.

	CLO	Cognitive Level	PLO Mapping
1	**Apply** software engineering concepts to construct (i.e. design, develop, and test) software in a team setting	3	3
2	**Implement** software design patterns as a part of software construction activity	3	3
3	**Design** test cases for a software system	6	3
4	**Use** a version control system as a part of software construction activity	3	5
5	**Implement** the deployment-related steps to bring the constructed software into use	3	3

Course Outcomes

The students will be able to:

- Convert OO design into code
- Write unit test cases in JUnit
- Understand various code smells and refactoring
- Understand selected GoF design patterns and their corresponding code

Grading Breakup	
Quizzes	10%
Class Participation	5%
Project	25%
Sessional I	12.5%
Sessional II	12.5%
Final Examination	35%

Plagiarism Policy

- Any proven academic dishonesty may lead to a straight ‘F’ in the course.

Course Schedule (Tentative)			
Week	**Theory**	**Deliverables**	**Lab**
1	a) Intro to the course b) Java basics: inheritance & polymorphism in Java, static/dynamic binding[1], Java interfaces[2], Wrapper classes for primitives	SRS (functional), UC diagram	
2	a) Java exception handling[3] (try, catch, throw, throws, finally), chained Exceptions[4]	DM, SSD	
3	a) Generics and Collections framework b) Reducing coupling: dependency injection, dependency inversion principle[5] c) Event-driven programming: events and event handlers, callbacks etc. AWT[6] d) MVC[7][8][9][10]	Screen Mockups, ER diagram	
4	Observer[11] design pattern and its use[12] Requirements elicitation for the project, Test-driven development Testing: Writing unit tests in JUnit5	UC#1 SD, refined DCD	
5	Unit testing contd., Logging, Git: concepts and commands		
6	Git contd.: merging techniques		
7	First Session Examinations		
8	Converting design to code: class diagrams, sequence[13]/collaboration diagrams, state machine diagrams (switches, State design pattern)		
9	MVP in code		

[1] Static and Dynamic Binding in Java | Baeldung
[2] https://chat.openai.com/share/1576802b-e2c6-4a11-95cc-e13ac28935ff
[3] Lesson: Exceptions - Java™ Tutorials
[4] Chained Exceptions in Java | Baeldung
[5] https://chat.openai.com/share/d8234900-c318-4a9e-af0a-b2cbad53d3f0
[6] Java AWT Tutorial for Beginners | AWT in Java GUI | Edureka
[7] A Swing Architecture Overview
[8] The Model-View-Controller Architecture - Java Swing [Book]
[9] Abstract Window Toolkit (AWT)
[10] The simplest Model View Controller (MVC) Java example
[11] The Observer Pattern in Java | Baeldung
[12] Advanced Uses of the Observer Pattern in Java
[13] SequenceDiagram.org

10	Architectures: layered[14][15][16][17], Using UML component[18] diagrams to represent layered architecture and applying dependency inversion[19] principle between layers.	Iteration#1 code	
11	Implementing Abstract Factory, Singleton and Facade[20] design patterns		
12	Second Session Examinations		
13	Code smells & refactoring		
14	Code smells & refactoring		
15	*Optional: Design patterns and their implementation in Java: Factory Method[21], Composite, Proxy, Decorator*	Iteration#2 code	
16	*Optional: Design patterns and their implementation in Java: State, Interpreter, Chain of Responsibility, Strategy* Delivering the project as an executable jar		

Textbook(s)/Supplementary Readings
• https://canvas.uw.edu/courses/1100150/pages/course-schedule

[14] Core J2EE Patterns - Data Access Object
[15] What is Three-Tier Architecture | IBM
[16] Design Patterns: Data Access Object
[17] 1. Layered Architecture - Software Architecture Patterns [Book]
[18] The component diagram - IBM Developer
[19] A Solid Guide to SOLID Principles | Baeldung
[20] Layered Architecture using the Facade Design Pattern
[21] Factory method for designing pattern - GeeksforGeeks