Identification	**Subject**	CMS 205 Object Oriented Programming, 3KU /6ECTS credits
	Department	Computer Science
	Program	Undergraduate
	Term	Fall, 2023
	Instructor	Javad Mehri-Tekmeh (PhD)
	E-mail:	jmehri@khazar.org
	Phone:	(+994 12) 421 1093 (ext. 266)
	Classroom/hours	41 Mehseti str. (Neftchilar campus), Thursday 17:00-18:30, Friday 17:00-18:30
Prerequisites	CMS 106 Fundamentals of Programming,	
Language	English	
Compulsory/Elective	Required	
Required textbooks and course materials	***Core Textbook:*** 1. Schildt, Herbert, and Dale John Skrien. 2013. Java Programming: A Comprehensive Introduction. New York: McGraw-Hill.. 2. Deitel, Paul J., and Harvey M. Deitel. 2018. Java How to Program. Early Objects. New York, NY: Pearson.	
Course website	This course combines traditional face-to-face classes.	
Course outline	This course serves as an intermediate-level exploration of Object-Oriented Programming (OOP) principles and Java programming for students who have prior programming experience in C or C++. Through hands-on exercises and projects, students will deepen their understanding of OOP concepts and develop proficiency in Java programming. The course includes a comprehensive project with five phases to reinforce learning and two exams to assess student progress.	
Course objectives	By the end of this course, students should be able to: • Understand the fundamentals of Object-Oriented Programming (OOP) principles. • Demonstrate proficiency in Java programming. • Apply OOP concepts, such as inheritance, polymorphism, encapsulation, and abstraction, in practical scenarios. • Design and implement object-oriented solutions to real-world problems. • Collaborate effectively on software development projects. • Debug and troubleshoot Java code. • Prepare for advanced Java programming and software development courses.	
Learning outcomes	By successfully completing this course, students will be able to demonstrate the following outcomes: • **Java Proficiency:** Develop a strong command of the Java programming language, including its syntax, data types, and standard libraries. • **Object-Oriented Programming:** Understand and apply fundamental Object-Oriented Programming (OOP) concepts, such as encapsulation, inheritance, polymorphism, and abstraction. • **Problem Solving:** Analyze and solve complex problems by designing and implementing object-oriented solutions using Java. • **Software Development:** Gain hands-on experience in software development by creating Java applications that follow industry best practices. • **Debugging Skills:** Develop the ability to debug and troubleshoot Java code effectively, identifying and rectifying errors. • **Project Management:** Collaborate with peers to complete a multi-	

phase software development project, applying project management skills and best practices.

- **File Handling:** Demonstrate proficiency in reading and writing files using Java's input/output (I/O) capabilities.
- **Graphical User Interfaces:** Create graphical user interfaces (GUIs) using Java Swing and JavaFX to design user-friendly applications.
- **Data Structures:** Implement and manipulate data structures provided by the Java Collections Framework, including lists, sets, and maps.
- **Concurrency:** Understand multithreading concepts in Java and apply synchronization techniques to ensure thread safety.
- **Advanced Topics:** Explore advanced Java topics, such as generics to write more efficient and maintainable code.
- **Software Design Principles:** Apply SOLID principles and design patterns to create well-structured and maintainable software.
- **Exam Readiness:** Prepare for and perform well in both the midterm and final exams, demonstrating knowledge retention and problem-solving skills.
- **Effective Communication:** Communicate technical concepts and ideas clearly and concisely, both in written and oral form.
- **Critical Thinking:** Develop critical thinking skills by evaluating and selecting appropriate solutions for programming challenges.
- **Self-Learning:** Cultivate the ability to continue learning ndependently, keeping up with advancements in Java and software development.
- **Ethical Programming:** Understand the ethical considerations in software development, including issues related to privacy, security, and intellectual property.
- **Teamwork:** Collaborate effectively in a team environment, demonstrating professionalism, communication skills, and the ability to meet project deadlines.

Teaching methods	**Lecture**		X
	Group discussion		X
	Experiential exercise		X
Evaluation	**Methods**	**Date/deadlines**	**Percentage (%)**
	Midterm Exam		25
	Project		35
	Final Exam		40
	Total		100

Policy

Project description

In this project, students will develop a graphical game in Java inspired by classic Atari-style games. The goal is to create an engaging and interactive gaming experience while focusing on object-oriented programming principles, concurrency for smooth gameplay, and the integration of sound effects to enhance the user experience. The project will be divided into several phases to help students build the game incrementally.

Game Concept:

The game concept is open-ended, but it should incorporate the following elements:

Graphics: The game must have a graphical interface that includes a player character (e.g., spaceship, paddle, character) and interactive objects/enemies (e.g., asteroids, aliens, obstacles).

Gameplay: Define clear objectives and rules for the game. The player should be able to control the character using keyboard input (e.g., arrow keys) or mouse interactions. The game should have scoring, levels, and a win/lose condition.

Concurrency: Implement concurrency in the game to handle multiple game elements simultaneously. For example, if there are multiple enemies or bullets on the screen, concurrency should ensure smooth movement and interactions.

Sound Effects: Integrate sound effects to make the game more immersive. Assign sounds for actions such as shooting, collision, level completion, and game over.

Project Phases:
Break the project into several phases to guide students through its development:

Phase 1: Game Design
- Define the game concept, objectives, and rules.
- Create a basic (UML) diagram outlining the game's object-oriented structure.
- Design the graphical elements (sprites) for the player character and interactive objects/enemies.
- Plan the game's graphical user interface (GUI) layout.

Phase 2: Game Initialization and Graphics
- Set up the game window and initialize the game environment.
- Implement the rendering of the player character and initial game elements.
- Allow basic player movement and interactions.

Phase 3: Game Logic and Concurrency
- Implement the game's core logic, including scoring, level progression, and collision detection.
- Introduce concurrency to handle multiple game elements.
- Test and debug the concurrency aspects.

Phase 4: Sound Effects Integration
- Incorporate sound libraries (e.g., Java Sound API) for playing sound effects.
- Assign appropriate sounds to in-game events.
- Test and fine-tune the sound effects.

Phase 5: Polish and Finalization
- Optimize the game for performance and user experience.
- Implement game over and win conditions.
- Add additional features or enhancements if time allows.

Final Presentation:
At the end of the project, each student or team should present their game to the class. They should explain their design choices, demonstrate gameplay, and discuss the

	challenges they faced during development. ▪ **Preparation for class** The structure of this course emphasizes the importance of independent study and preparation outside of class. The lecture material will concentrate on the key points raised in the text. Reading the assigned chapters and becoming acquainted with them prior to class will aid your understanding of the lecture. Following the lecture, you should review your notes and work on relevant problems and cases from the chapter's end, as well as sample exam questions. We will also have many review sessions throughout the semester. These review sessions will take place during the regular class times. ▪ **Withdrawal (pass/fail)** This course strictly adheres to the grading policy of the School of Engineering and Applied Science. As a result, a student is normally expected to pass with a grade of at least 60%. In the event of failure, he or she will be required to repeat the course the following term or year. ▪ **Cheating/plagiarism** Cheating or other plagiarism during the Quizzes, Mid-term and Final Examinations will lead to paper cancellation. In this case, the student will receive a zero (0) without further consideration. ▪ **Professional behavior guidelines** During class, students must act in a way that fosters a positive academic and professional environment. Unauthorized conversations and unethical behavior are forbidden. ▪ **Ethics** Students should not arrive in late to class. All cell phones must be turned off and stowed away before entering class. Use of any electronic devices is not allowed in the classroom and violators will be punished accordingly.

Tentative Schedule

Week	**Date/Day (tentative)**	**Topics**	**Textbook**
1	21-Sep, 22-Sep	**Introduction to Java and OOP Principles** • Course overview and expectations • Introduction to Java and its history • Basic OOP concepts: classes and objects **Java Development Environment Setup** • Installing Java Development Kit (JDK) and Integrated Development Environment (IDE) • Writing and running a simple Java program **Project Description**	**Ch. 1**
2	28-Sep, 29-Sep	**Variables and Data Types in Java** • Primitive data types • Variables, constants, and naming conventions • Typecasting and conversions **Operators and Expressions** • Arithmetic, relational, logical, and assignment operators • Expressions and precedence rules	**Ch. 2**
3	5-Oct, 6-Oct	**Control Structures in Java** • Conditional statements: if, else-if, switch	**Ch. 3, 5**

		• Looping constructs: for, while, do-while **Arrays and Strings** • Declaring and manipulating arrays • String manipulation in Java	
4	12-Oct, 13-Oct	**Object-Oriented Programming Concepts** • Classes and objects in detail • Constructors and methods • Encapsulation and access modifiers **Inheritance and Polymorphism** • Inheritance and its advantages • Polymorphism and method overriding	**Ch. 4, 6-7**
5	19-Oct, 20-Oct	**Phase 1 due: Game Design** **Abstract Classes and Interfaces** • Abstract classes and methods • Interfaces and multiple inheritance **Exception Handling** • Understanding exceptions • Try-catch blocks and exception propagation	**Ch. 8, 10**
6	26-Oct, 27-Oct	**GUI Programming with Swing** • Introduction to Java Swing • Creating graphical user interfaces	**Ch. 17-20**
7	2-Nov, 3-Nov	**Phase 2 due: Game Initialization and Graphics** **File Handling in Java** • Reading and writing files in Java • Working with streams	**Ch. 11**
8	9-Nov, 10-Nov	**Collections Framework** • Overview of Java Collections Framework • Lists, sets, and maps	**Ch. 13**
9	16-Nov, 17,-Nov	**Review for Midterm exam & Exercise.** **Midterm Exam.**	
10	23-Nov, 24-Nov	**Threads and Concurrency** • Multithreading in Java • Synchronization and thread safety	**Ch. 12**
11	30-Nov, 1-Dec	**Generics** • Introduction to generics • Generic classes and methods	**Ch. 14**
12	7-Dec, 8-Dec	**Phase 3 due: Game Logic and Concurrency** **Introduction to JavaFX** • Basics of JavaFX • Building JavaFX applications **Event Handling in JavaFX** • Handling events in JavaFX applications • Event-driven programming	**[2] Ch. 12-13**

13	14-Dec, 15-Dec	**JavaFX Graphics and Multimedia**	**[2] Ch. 22**
14	21-Dec, 22-Dec	**Phase 4 due: Sound Effects Integration** **Review and Project Demonstration**	
15	28-Dec, 29-Dec	**Phase 5 due: Polish and Finalization** **Final Exam Review**	
	TBA	**Final Exam**	

This syllabus is a guide for the course and any modifications to it will be announced in advance.