School of Business and Management Sciences / Management Information Systems (English)

2023 - 2024 Academic Year

OBJECT ORIENTED PROGRAMMING

Syllabus

Course Description					
Name	**Code**	**Semester**	**T+A Hour**	**Credit**	**ECTS**
OBJECT ORIENTED PROGRAMMING	MIS3212179	Spring Semester	3+0	3	4

Prerequisites Courses	
Recommended Elective Courses	
Language of Instruction	English
Course Level	First Cycle (Bachelor's Degree)
Course Type	Required
Course Coordinator	Assist.Prof. Kevser ŞAHİNBAŞ
Name of Lecturer(s)	Assist.Prof. Kevser ŞAHİNBAŞ, Lect. Nada A. M. MISK
Assistant(s)	
Aim	The aim of this course is to develop an understanding of the principles of object oriented programming so that students can apply it in IT projects.
Course Content	This course contains; Genealogy of object oriented languages: structured programming, procedural programming,Abstract data types, encapsulation,Typed and untyped languages, Coupling and cohesion,Encapsulation. Classes and objects. Class members: Data members (fields) and member functions (methods). Class member visibility (private, public, protected). Class variables and instance variables. Class methods and instance methods. Service methods and support methods. ,Class hierarchies. Single and multiple inheritance. Inter-class relationships.,Memory management. Garbage collection. Methods and messages. Method signatures. Method and operator overloading. Method overriding. Abstract classes. Dynamic (late) binding. Polymorphism. Software reuse. Subclasses (derived classes). Superclasses (base classes). Invocation of superclass methods and constructors.,Objects vs. variables. Classes vs. types. Delegation. Collection classes. Class libraries.,Unified Modelling Language (UML). Use case diagrams: actors, system boundary, --uses-- and --extends--,Scenarios. Class diagrams: associations, aggregation, dependency, and inheritance. Object interaction diagrams, object state transition diagrams.,Object constraint language (OCL):,Design patterns. Pattern documentation: motivation, prerequisites, structures, participants and consequences. Examples of patterns: Adapter, Decorator, Iterator, Observer, Singleton,In class example and project picking,Implementation of designs in an object-oriented programming language,Testing object oriented code. Class testing, constructing class tests from OCL or state transition diagrams, test driver construction. Testing interactions and class hierarchies.

Course Learning Outcomes	**Teaching Methods**	**Assessment Methods**
1. will be able to se Eclipse as an Integrated Development Environment.	16, 6, 9	A
1.1. Gets enough information about Eclipse.	6, 9	A
1.2. Uses Eclipse knowing why and how to use .	16, 6	A
2. Will be able to apply standards and principles to write truly readable code.	6, 9	A
2.1. Defines readable codes.	6, 9	A
2.2. Knows what are the standards principles for readable codes, and fulfill these standards and principles in the applications.	6, 9	A
3. will be able to produce class diagrams, object interaction diagrams and object state transition diagrams for a given problem	6, 9	A
3.1. Produces class diagrams for a given problem.	6, 9	A
3.2. Produces object interaction diagrams for a given problem.	6, 9	A
3.3. Produces object state transition diagrams for a given problem.	6	A
4. will be able to describe the essential features of an object-oriented programming language	6, 9	A
4.1. Defines basic principles of the object oriented programming language.	6	A
4.2. Applies the facilitator side of the object oriented programming language.	6, 9	A
5. will be able to produce and debug code fragments that illustrate principles of object oriented software development.	6, 9	A
5.1. Generates code fragments based on object-oriented software development principles.	6, 9	A
5.2. Extracts errors according to object-oriented software development principles.	6, 9	A
6. Will be able to define the fundamentals of input and output using the java.io library.	16, 6, 9	A
6.1. Uses the java.io library effectively .	6, 9	A
6.2. Defines input and output bases.	6	A
7. will be able to define the concepts of object-oriented design, polymorphism, information hiding, and inheritance.	16, 6, 9	A
7.1. Defines the concept of object oriented design.	6	A
7.2. Defines the concept of object oriented polymorphism.	6, 9	A
7.3. Defines concept of object oriented information storage.	6	A
7.4. Defines the concept of object oriented inheritance.	6	A

Teaching Methods	16: Question - Answer Technique, 6: Experiential Learning, 9: Lecture Method
Assessment Methods	A: Traditional Written Exam

Lecture Schedule

Sequence	**Topics**	**Preliminary Preparation**
1	Genealogy of object oriented languages: structured programming, procedural programming	
2	Abstract data types, encapsulation	
3	Typed and untyped languages, Coupling and cohesion	
4	Encapsulation. Classes and objects. Class members: Data members (fields) and member functions (methods). Class member visibility (private, public, protected). Class variables and instance variables. Class methods and instance methods. Service methods and support methods.	
5	Class hierarchies. Single and multiple inheritance. Inter-class relationships.	

School of Business and Management Sciences / Management Information Systems (English)

2023 - 2024 Academic Year

OBJECT ORIENTED PROGRAMMING

Syllabus

Lecture Schedule		
Sequence	**Topics**	**Preliminary Preparation**
6	Memory management. Garbage collection. Methods and messages. Method signatures. Method and operator overloading. Method overriding. Abstract classes. Dynamic (late) binding. Polymorphism. Software reuse. Subclasses (derived classes). Superclasses (base classes). Invocation of superclass methods and constructors.	
7	Objects vs. variables. Classes vs. types. Delegation. Collection classes. Class libraries.	
8	Unified Modelling Language (UML). Use case diagrams: actors, system boundary, --uses-- and --extends--	
9	Scenarios. Class diagrams: associations, aggregation, dependency, and inheritance. Object interaction diagrams, object state transition diagrams.	
10	Object constraint language (OCL):	
11	Design patterns. Pattern documentation: motivation, prerequisites, structures, participants and consequences. Examples of patterns: Adapter, Decorator, Iterator, Observer, Singleton	
12	In class example and project picking	
13	Implementation of designs in an object-oriented programming language	
14	Testing object oriented code. Class testing, constructing class tests from OCL or state transition diagrams, test driver construction. Testing interactions and class hierarchies	

Evaluation Methods	**Weight(%)**
Midterm Exam	40
General Exam	60

Resources

Pearson Platform (my programming lab will be used as the main platform for e-text and exercises):
Introduction to Java Programming and Data Structures –Y. Daniel Liang, Twelfth Edition, Pearson (https://mlm.pearson.com/northamerica/myprogramminglab/)
Reading list:

1. Java How to Program, Deitel & Deitel, Prentice Hall, 9th Edition, 2012
2. Thinking in java, Bruce Eckel, Prentice Hall, 4th edition 2006
3. Java 2 The Complete Reference, Herbert Schildt, McGraw Hill, 7th Ed.2007
4. Java programming: A beginners Guide to learning Java, step by step. By Troy Dimes 2015
5. Java: A beginner's Guide. By Herbert Schildt 2014
6. Head First Java, 2nd Edition By Katy Sierra and Bert Bates 2005
7. Learning Java 4th edition By Patrick Niemeyer, Daniel Leuck. Publisher: O'Reilly Media, 2013
8. Java object oriented language, M. Smith, McGraw Hill 1999.
9. An Introduction to object programming with Java, McGraw Hill, 5th Ed.1999