هيئة تقويم التعليم والتدريب
Education & Training Evaluation Commission

اعتماد
NCAAA
T4
2020

Course Specifications

Course Title:	**Programming Languages**
Course Code:	**503PMAI-3**
Program:	**Professional Master of Artificial Intelligence**
Department:	**Computer Science**
College:	**Computer Science and information systems**
Institution:	**Najran University**

Table of Contents

A. Course Identification

1. Credit hours:3							
2. Course type							
a.	University □	College √	Department √	Others □			
b.	Required √	Elective □					
3. Level/year at which this course is offered: 2nd level/ 1st year							
4. Pre-requisites for this course (if any)**:**							
5. Co-requisites for this course (if any)**: NA**							

6. Mode of Instruction (mark all that apply)

No	Mode of Instruction	Contact Hours	Percentage
1	Traditional classroom	50	100%
2	Blended		
3	E-learning		
4	Distance learning		
5	Other		

7. Contact Hours (based on academic semester)

No	Activity	Contact Hours
1	Lecture	30
2	Laboratory/Studio	20
3	Tutorial	
4	**Others** (specify)	
	Total	50

B. Course Objectives and Learning Outcomes

1. Course Description:

This course describes history of programming languages, formal models for specifying languages, design goals, run-time structures, and implementation techniques, along with a survey of principal programming language paradigms.

2. Course Main Objective

After successful completion of this course students should be able to:

use advanced programming techniques to solve computing problems. These include but are not limited to: polymorphism, inheritance, abstract classes, interfaces enumerated data types exceptions file I/O recursion data structures such as multi-dimensional arrays, ArrayList, HashTable, linked lists use appropriate object oriented design techniques. understand UML diagrams and their relationship to the design process. use appropriate testing techniques to thoroughly test an application during development. understand contiguous and linked implementation of stacks and queues. read and understand software specifications to implement code that conforms to the specifications and to course coding standards.

3. Course Learning Outcomes

CLOs		Aligned PLOs
1	**Knowledge and Understanding**	
1.1	understand UML diagrams and their relationship to the design process.	K1
1.2	understand contiguous and linked implementation of stacks and queues.	K3
1.3	read and understand software specifications to implement code that conforms to the specifications and to course coding standards	K2
1...		
2	**Skills**	
2.1	use advanced programming techniques to solve computing problems. These include but are not limited to: polymorphism, inheritance, abstract classes, interfaces enumerated data types exceptions file I/O recursion data structures such as multi-dimensional arrays, ArrayList, HashTable, linked lists	S2
2.2	use appropriate object-oriented design techniques.	S1
2.3	use appropriate testing techniques to thoroughly test an application during development.	S3
2.4		
2.5		
3	**Competences:**	
3.1		
3.2		
3.3		
3...		

C. Course Content

No	List of Topics	Contact Hours
1	Specification of programming languages: Syntax o Semantics □ Operational Semantics □ Denotational Semantics □ Axiomatic Semantics □ Attribute Grammars	3.5
2	Specification of programming languages o Syntax o Semantics □ Operational Semantics □ Denotational Semantics □ Axiomatic Semantics □ Attribute Grammars	3.5
3	Specification of programming languages o Syntax o Semantics □ Operational Semantics □ Denotational Semantics □ Axiomatic Semantics □ Attribute Grammars	3.5
4	Issues in language design o Names, scope, and binding o Types o Control Flow o Control Abstractions	3.5
5	Issues in language design o Names, scope, and binding o Types o Control Flow o Control Abstractions	3.5

6	Issues in language design o Names, scope, and binding o Types o Control Flow o Control Abstractions	3.5
7	Issues in language design o Names, scope, and binding o Types o Control Flow o Control Abstractions	3.5
8	Issues in language design o Names, scope, and binding o Types o Control Flow o Control Abstractions	3.5
9	Programming language paradigms o Data abstraction and object-oriented programming (examples: Java, Smalltalk, C++) o Non-imperative paradigms □ Functional languages (examples: Scheme, ML, Haskell) □ Logic programming (example: Prolog) o Dynamic and scripting languages (examples: lua, csh, Python, Ruby, Perl, tcl, etc.) o Concurrent programming (examples: Java, SR, OpenMP)	3.5
10	Programming language paradigms o Data abstraction and object-oriented programming (examples: Java, Smalltalk, C++) o Non-imperative paradigms □ Functional languages (examples: Scheme, ML, Haskell) □ Logic programming (example: Prolog) o Dynamic and scripting languages (examples: lua, csh, Python, Ruby, Perl, tcl, etc.) o Concurrent programming (examples: Java, SR, OpenMP)	3.5
11	Programming language paradigms o Data abstraction and object-oriented programming (examples: Java, Smalltalk, C++) o Non-imperative paradigms □ Functional languages (examples: Scheme, ML, Haskell) □ Logic programming (example: Prolog) o Dynamic and scripting languages (examples: lua, csh, Python, Ruby, Perl, tcl, etc.) o Concurrent programming (examples: Java, SR, OpenMP)	3.5
12	Programming language paradigms o Data abstraction and object-oriented programming (examples: Java, Smalltalk, C++) o Non-imperative paradigms □ Functional languages (examples: Scheme, ML, Haskell) □ Logic programming (example: Prolog) o Dynamic and scripting languages (examples: lua, csh, Python, Ruby, Perl, tcl, etc.) o Concurrent programming (examples: Java, SR, OpenMP)	3.5
13	Programming language paradigms o Data abstraction and object-oriented programming (examples: Java, Smalltalk, C++) o Non-imperative paradigms □ Functional languages (examples: Scheme, ML, Haskell) □ Logic programming (example: Prolog) o Dynamic and scripting languages (examples: lua, csh, Python, Ruby, Perl, tcl, etc.) o Concurrent programming (examples: Java, SR, OpenMP)	3.5
14	Programming language paradigms o Data abstraction and object-oriented programming (examples: Java, Smalltalk, C++) o Non-imperative paradigms □ Functional languages (examples: Scheme, ML, Haskell) □ Logic programming (example: Prolog) o Dynamic and scripting languages (examples: lua, csh, Python, Ruby, Perl, tcl, etc.) o Concurrent programming (examples: Java, SR, OpenMP)	3.5
15	Programming language paradigms o Data abstraction and object-oriented programming (examples: Java, Smalltalk, C++) o Non-imperative paradigms □ Functional languages (examples: Scheme, ML, Haskell) □ Logic programming (example: Prolog) o Dynamic and scripting languages (examples: lua, csh, Python, Ruby, Perl, tcl, etc.) o Concurrent programming (examples: Java, SR, OpenMP)	3.5
Total		50

D. Teaching and Assessment

1. Alignment of Course Learning Outcomes with Teaching Strategies and Assessment Methods

Code	Course Learning Outcomes	Teaching Strategies	Assessment Methods
1.0	**Knowledge and Understanding**		
1.1	understand UML diagrams and their relationship to the design process.	TS-1: Relate Course Learning Outcomes (CLOs) to the topics TS-2: Giving Lectures in PPT, recalling the lecture through asking Questions. Clarifying doubts on Lecture. TS-3: Conducting a discussion of real life problems, among teacher, students	Quiz Assignments Midterm Examination Final Examination
1.2	understand contiguous and linked implementation of stacks and queues.		
…	read and understand software specifications to implement code that conforms to the specifications and to course coding standards		
2.0	**Skills**		
2.1	use advanced programming techniques to solve computing problems. These include but are not limited to: polymorphism, inheritance, abstract classes, interfaces enumerated data types exceptions file I/O recursion data structures such as multi-dimensional arrays, ArrayList, HashTable, linked lists	TS-1: Relate Course Learning Outcomes (CLOs) to the topics TS-2: Giving Lectures in PPT, recalling the lecture through asking Questions. Clarifying doubts on Lecture. TS-3: Conducting a discussion of real life problems, among teacher, students TS-4: Cooperative learning among the students. Encourage students to browse different journals, seminars or websites at their leisure time to have a better understanding about the course	Quiz Assignments Midterm Examination Final Examination,
2.2	use appropriate object oriented design techniques.		Quiz, Assignments Final Examination
2.3	use appropriate testing techniques to thoroughly test an application during development.		Quiz Assignments Final Examination
2.4			Lab Assignments, Midterm Examination,
2.5			Final Examination
3.0	**Competences**		
3.1			
3.2			
…			

2. Assessment Tasks for Students

#	Assessment task*	Week Due	Percentage of Total Assessment Score
1	Quiz1	3rd week	5%
2	Midterm 1	6th week	20%
3	Project	5th week	15%
4	Theory Assignments	2th , 5th , 8th , 10th weeks	5%

#	Assessment task*	Week Due	Percentage of Total Assessment Score
5	Lab Assignments	7th week	10%
6	Quiz2	10th week	5%
8	Final Exam	12th or 13th week	40%

***Assessment task** (i.e., written test, oral test, oral presentation, group project, essay, etc.)

E. Student Academic Counseling and Support

Arrangements for availability of faculty and teaching staff for individual student consultations and academic advice :

- Weekly office hours + Appointments
- Weekly academic advising hours
- Extra weekly 2 office hours prior to exams.
- Tutorials are also provided to the students

F. Learning Resources and Facilities

1.Learning Resources

Required Textbooks	Starting Out with Java: From Control Structures through Objects, 4/E. Tony Gaddis, Addison-Wesley, 2010.
Essential References Materials	
Electronic Materials	
Other Learning Materials	

2. Facilities Required

Item	Resources
Accommodation (Classrooms, laboratories, demonstration rooms/labs, etc.)	Room B-58 Laboratory A-16L
Technology Resources (AV, data show, Smart Board, software, etc.)	Data show, PCs.
Other Resources (Specify, e.g. if specific laboratory equipment is required, list requirements or attach a list)	• Printer is important in the lab to print reports and some snapshots. • Projector and PC for the lab instructor is required

G. Course Quality Evaluation

Evaluation Areas/Issues	Evaluators	Evaluation Methods
Online course survey	Students	Indirect
Focus group discussion with small groups of students.	Instructor	Direct
Extent of achievement of course learning outcomes	instructor	Direct

Evaluation areas (e.g., Effectiveness of teaching and assessment, Extent of achievement of course learning outcomes, Quality of learning resources, etc.)

Evaluators (Students, Faculty, Program Leaders, Peer Reviewer, Others (specify)

Assessment Methods (Direct, Indirect)

H. Specification Approval Data

Council / Committee	Computer Science Department Council
Reference No.	14440203-0185-00002
Date	1st Sep, 2022