2023
TP-153

Course Specification
(Bachelor)

Course Title:	**Object Oriented Programming**
Course Code:	**CSC 1201**
Program:	**Bachelor in Computer Science**
Department:	**Computer Science**
College:	**Faculty of Computers and Information Technology**
Institution:	**University of Tabuk**
Version:	**1.0**
Last Revision Date:	**27 July 2022**

Table of Contents

A. General information about the course:

1. Course Identification

1. Credit hours: (4)					
2. Course type					
A.	☐University	☒ College	☐ Department	☐Track	☐Others
B.	☒ Required		☐Elective		
3. Level/year at which this course is offered: (Level 3/Year 2)					
4. Course general Description:					
This course introduces advanced programming skills and focuses on the core concepts of object-oriented programming. This course focuses on the understanding and practical mastery of object-oriented concepts such as classes, objects, data abstraction, inheritance and polymorphism.					
5. Pre-requirements for this course (if any)**:**					
CSC 1103					
6. Co-requisites for this course (if any)**:**					
N/A					
7. Course Main Objective(s):					
After completing this course, the students will be able to: **- Create and use classes and objects.** **- Declare and create an array of objects.** **- Understand inheritance and software reusability.** **- Design and implement systems that are easily extensible and maintainable with polymorphism.** **- Understand structures, inner classes and exceptions handling.** **- Understand how to create, read, write, and update files.**					

2. Teaching mode (mark all that apply)

No	Mode of Instruction	Contact Hours	Percentage
1	Traditional classroom	**75**	**100 %**
2	E-learning	-	-
3	Hybrid • Traditional classroom • E-learning	-	-

No	Mode of Instruction	Contact Hours	Percentage
4	Distance learning	-	-

3. Contact Hours (based on the academic semester)

No	Activity	Contact Hours
1.	**Lectures**	**45**
2.	**Laboratory/Studio**	**30**
3.	**Field**	**-**
4.	**Tutorial**	**-**
5.	**Others (specify)**	**-**
Total		**75**

B. Course Learning Outcomes (CLOs), Teaching Strategies and Assessment Methods

Code	Course Learning Outcomes	Code of PLOs aligned with program	Teaching Strategies	Assessment Methods
1.0	**Knowledge and understanding**			
1.1	**Identify the elements and principles of object-oriented programming with examples.**	**K1, K2, K3**	**Lectures, class discussions, lab discussions**	**Exams, lab assignments**
1.2	**Describe the concepts of object-oriented to software design**	**K1, K2, K3**		
1.3	**Describe the importance of programming, testing, and implementation principles.**	**K2,K3,K4**		
2.0	**Skills**			
2.1	**Design and develop object-oriented computer programs**	**S2, S3, S4**	**Lectures, class discussions, lab discussions**	**Exams, lab assignments**
2.2	**Design problems as steps so as to be solved systematically**	**S2**		
3.0	**Values, autonomy, and responsibility**			
3.1	**Communicate and work (effectively, ethically, and professionally) (individually and in groups/teamwork) to accomplish all the**	**V2**	**Lab discussions**	**Lab assignments**

Code	Course Learning Outcomes	Code of PLOs aligned with program	Teaching Strategies	Assessment Methods
	assigned duties and projects.		**Lab discussions**	**Lab works**

C. Course Content

No	List of Topics	Contact Hours
1.	**Classes and Objects – Part 1** - Describe objects and classes, and use classes to model objects. - Use UML graphical notation to describe classes and objects. - Demonstrate how to define classes and create objects. **Lab Experiment:** Implements the concepts of class and object creation.	5
2.	**Classes and Objects – Part 2** - Create objects using constructors. - Access objects via object reference variables. - Access an object's data and methods using the object member access operator (.). - Distinguish between object reference variables and primitive data type variables. - Distinguish between instance and static variables and methods. **Lab Experiment:** Implements the concepts of constructors, instance members, and static members.	5
3.	**Classes and Objects – Part 3** - Define private data fields with appropriate get and set methods. - Store and process objects in arrays. - Determine the scope of variables in the context of a class. - Use the keyword this to refer to the calling object itself. **Lab Experiment:** Implements the concepts of encapsulation and array of objects.	5

4.	**Thinking in Objects – Part 1**
	- Discover the relationships between classes.
	  ○ Association
	  ○ Aggregation
	  ○ Composition
	- Design programs using the object-oriented paradigm.
	Lab Experiment: Implements classes, objects, members of a class, and relationships among them.
		5

5.	**Thinking in Objects – Part 2** - Use the BigInteger and BigDecimal classes for computing very large numbers with arbitrary precisions - Use the String class to process immutable strings. - Use the StringBuilder and StringBuffer classes to process mutable strings. **Lab Experiment:** Implement programs using BigInteger, BigDecimal, String,	5

	StringBuilder, and StringBuffer classes.	
6.	**Inheritance – Part 1** - Define a subclass from a superclass through inheritance. - Invoke the superclass's constructors and methods using the super keyword. - Override instance methods in the subclass. **Lab Experiment:** Implements the concepts of Inheritance.	5
7.	**Inheritance – Part 2** - Distinguish differences between overriding and overloading. - Explore the toString() method in the Object class. **Lab Experiment:** Implement inheritance and show method overriding, and demonstrate method overloading.	5
8.	**Polymorphism – Part 1** - Discover polymorphism and dynamic binding. - Describe casting and explain why explicit downcasting is necessary. - Explore the equals method in the Object class. **Lab Experiment:** Implement the concept of polymorphism and dynamic binding.	5
9.	**Polymorphism – Part 2** - Store, retrieve, and manipulate objects in an Array. - Enable data and methods in a superclass accessible from subclasses using the protected visibility modifier. - Prevent class extending and method overriding using the final modifier. **Lab Experiment:** Implements the concepts of polymorphism and array of objects.	5
10.	**Inner Classes** - Define and use inner classes. - Private inner class and static inner class. - Access outer class from inner class. **Lab Experiment:** Implements the concept of inner classes.	5
11.	**Abstraction** - Design and use abstract classes. - Define abstract classes and define classes that extend abstract classes - Explore the similarities and differences among concrete classes, and abstract classes. **Lab Experiment:** Implements the concepts of abstract classes and abstract methods.	5
12.	**Interface** - Define and use interfaces and define classes that implement interfaces. - Explore the similarities and differences among concrete classes, abstract classes, and interfaces. **Lab Experiment:** Implements the concepts of interfaces.	5
13.	**Exception Handling – Part 1** - Get an overview of exceptions and exception handling. - Explore the advantages of using exception handling. - Distinguish exception types: Error vs. Exception and checked vs. unchecked. - Declare exceptions in a method header. **Lab Experiment:** Implements the concepts of exceptions.	5

14.	**Exception Handling – Part 2** - Throw exceptions in a method. - Write a try-catch block to handle exceptions. - Use the finally clause in a try-catch block. - Define custom exception classes. **Lab Experiment:** Implements the concepts of exception handling.	**5**
15.	**File Handling** - Discover file/directory properties. - Delete and rename files/directories. - Create files/directories. - Write data to a file. - Read data from a file. - Develop a program that replaces text in a file. **Lab Experiment:** Implements the concepts of file handling.	**5**
Total		**75**

D. Students Assessment Activities

No	Assessment Activities *	Assessment timing (in week no)	Percentage of Total Assessment Score
1.	Lab Assignments	2 – 14	20%
2.	Mid-Term Exam 1	7	15%
3.	Mid-Term Exam 2	12	15%
4.	Lab Exam	15	10%
5.	Final Exam	16	40%

*Assessment Activities (i.e., Written test, oral test, oral presentation, group project, essay, etc.).

E. Learning Resources and Facilities

1. References and Learning Resources

Essential References	Intro to Java Programming, Comprehensive Version, Y. Daniel Liang, Pearson, 12rd edition, 2019, ISBN 978-0136520153
Supportive References	Visual C# How to Program, Paul Deitel and Harvey Deitel, Pearson, 6th edition, 2016, ISBN 978-0134601540 C++ How to Program, Harvey M. Deitel and Paul J. Deitel, Pearson, 10th edition, 2016, , ASIN: 0134448235 Programming Languages Academy, Python for Beginners: 2 Books in 1: Python Programming for Beginners, Independently published, 2020, ISBN-10 : 1654414018, ISBN-13 : 978-1654414016
Electronic Materials	https://www.w3schools.com/java/default.asp
Other Learning Materials	Java How to Program, Late Objects, Paul Deitel and Harvey Deitel, Pearson,

	11rd edition, 2017, ISBN 978-0134791401 **Java 9 for Programmers, Paul Deitel and Harvey Deitel, Pearson, 4rd edition, 2017, ASIN: B071S84XCK**

2. Required Facilities and equipment

Items	Resources
facilities (Classrooms, laboratories, exhibition rooms, simulation rooms, etc.)	**Classroom**
Technology equipment (projector, smart board, software)	**Data show**
Other equipment (depending on the nature of the specialty)	**Students should have laptops**

F. Assessment of Course Quality

Assessment Areas/Issues	Assessor	Assessment Methods
Effectiveness of Teaching	Faculty, Program Leaders, and Advisory Board	Both Direct and Indirect
	Students	Indirect
Effectiveness of Students Assessment	Faculty, Program Leaders, Advisory Board, and Independent Opinion	Both Direct and Indirect
Quality of Learning Resources	Faculty, Students, and Advisory Board	Indirect
The Extent to which CLOs have been Achieved	Faculty, Program Leaders, Advisory Board, and Independent Opinion	Direct (as in section B) and Indirect/Surveys
	Students	Indirect
Other	-	-

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

Assessment Methods (Direct, Indirect)

G. Specification Approval

COUNCIL /COMMITTEE	
REFERENCE NO.	
DATE	