JSC «Kazakh-British Technical University»
Faculty of Information Technology

APPROVED BY

Dean of FIT
Hajiyev F. A.
_____
«____»_____ 2019.

SYLLABUS

Discipline: CSE212 Object-Oriented programming and Design
Number of credits: 3 (2/0/1)
Term: Fall 2019
Instuctor's full name:

Personal Information about the Instructor	Time and place of classes		Contact information	
	Lessons	Office Hours	Tel.:	e-mail
Shamoi Pakizar Associate Professor	According to the schedule	According to the schedule	87013490001	pakita.kbtu@gmail.com

Course duration: 3 credits, 15 weeks (45 class hours)
Course pre-requisites: Programming Principles 2

Course Description:

Object Technology has been in development for over forty years. It is now embedded in such diverse areas as requirements engineering, software architecture, analysis, design, programming, testing, deployment and maintenance. The fundamental concepts of object oriented programming will be studied using the Java programming language. The main objective of the course is to develop an understanding of the principles underpinning object oriented programming and apply object-based approaches using Java programming language. You will discover how to create flexible and reusable software, by applying object-oriented design principles and guidelines. And, you will be able to communicate these designs in a visual notation known as Unified Modelling Language (UML).

Course Goals:

During the course student will obtain the knowledge of fundamental principles of Object-Oriented programming. In particular, he will become familiar with the concepts of classes, objects, interfaces, inheritance, nested and abstract classes, polymorphism and data encapsulation, design patterns, UML. Moreover, he will master basic Java constructs – java collections (sets, lists, maps), exceptions, input / output, java networking facilities, etc. The main goal to be achieved during this course is to be able to use a high-level object-oriented programming language as a problem-solving tool - including basic data structures and algorithms, object-oriented programming techniques, and software documentation.

Literature:

Required:

1. Pakizar Shamoi. Object-oriented Programming and Design. KBTU, 2013

Supplementary:

2. Java API documentation: https://docs.oracle.com/javase/7/docs/api/index.html?overview-summary.html
3. Object-oriented Software Engineering: Practical Software Development Using UML and Java (Book by Robert Laganière and Timothy C. Lethbridge)
4. H. M. Deitel. Java™ How to Program, Sixth Edition. Prentice Hall, 2004.
5. Ivor Horton. Beginning Java™ 2, JDK™ 5 Edition. Wiley Publishing, 2005.
6. Anban Pillay. Object Oriented Programming using Java. School of Computer Science, 2007
7. Fowler M., UML Distilled, Addison-Wesley (3rdEd).
8. Weisfeld M., The Object-Oriented Thought Process, Addison-Wesley Professional (4th Ed.), 2013
9. Shalloway A., Trott J., Design Patterns Explained: A New Perspective on Object-oriented Design,AddisonWesley (2nd Ed), 2004
10. David J.Eck. Introduction to Programming Using Java. Department of Mathematics and Computer Science, Hobart and William Smith Colleges, 2006.

COURSE CALENDAR

Week	Class work					SIS (students independent study)		TSIS (teacher supervised independent study)	
	Topic	Seminars, hours	Lab, hours	Lectures, hours	Chapters for reading	Hours	Description	Hours	Description
1	**Lecture #1. Object Orientation as a New Paradigm . Introduction to Java programming language.** Java features. JVM. JRE. Java syntax. Java Api. Variables. Data types used in Java. Control statements. OO Approach. Genealogy of object oriented languages. Comparison to procedural programming.	1	0	2	Ch [1],[2]	1	Lab #1,	3	
2	**Lecture #2. Fundamentals of Objects and Classes.** Class members and instance members. Access control. Creating objects. Methods. Constructors. Initialization	1	0	2	Ch [3]	1	Lab #1	3	

	blocks. Methods overloading.								
3	**Lecture #3. Fundamentals of Objects and Classes.** Packages. Naming conventions. Class importation. Class design. Java API and Core Java classes	1	0	2	Ch [3]	1	Lab #2, SIS1	3	
4	**Lecture #4. Inheritance, Polymorphism and Abstract classes.** Relationships among classes. Extending existing classes. Inheritance and class hierarchy.	1	0	2	Ch [4]	1	Lab #2, SIS1	3	TSIS 1
5	**Lecture #5. Inheritance, Polymorphism and Abstract classes.** Methods overloading and overriding . Subclass and Superclass. Type Conversion. Polymorhism	1	0	2	Ch [4]	1	Lab #3	3	
6	**Lecture #6. Inheritance, Polymorphism and Abstract classes.** Abstract classes and methods. Special variables *this* and *super*. Design hints for inheritance.	1	0	2	Ch [4]	1	Lab #3	3	
7	**Lecture #7. Interfaces** Interface members. Extending interfaces. Object cloning. Interfaces and abstract classes.	1	0	2	Ch [5]	1	SIS2	3	
8	**Lecture #8. Interfaces** Marker interfaces. Cloneable and Comparable interfaces. Nested classes. Difference between Interfaces and Abstract classes.	1	0	2	Ch[5]	1	Lab#4	3	TSIS2
9	**Lecture #9. Collections & Data Structures** HashSet. TreeSet. HashMap. LinkedList. Lists. Vector. Stack. Iteration. Sorting a collection. Implementing user-defined collections.	1	0	2	Ch [7]	1	Lab#5	3	

	Viewing and analyzing Collections hierarchy, demonstrating OOP principles realization.								
10	**Lecture #10. Files and streams. Exceptions** Streams, Readers and Writers. PrintWriter. Scanner. RandomAccessFile, Buffered Streams.InputStream/OutputStream. Reader/Writer. *Viewing and analyzing Exceptions hierarchy, demonstrating OOP principles realization.* Checked & Unchecked exceptions. Claiming Exceptions. Throwing Exceptions . Catching Exceptions Handling the exceptions. try-catch block, finally clause. *Viewing and analyzing Streams hierarchy, demonstrating OOP principles realization.*	1	0	2	Ch [8],[9]	1	SIS3, Lab #5, #6	3	Assignments of Student Project
11	**Lecture #11. Threads** Creating and executing threads. Managing threads. Thread synchronization. Life-cycle of a thread. Thread priorities. Thread Groups. *Viewing and analyzing Threads hierarchy, demonstrating OOP principles realization.*	1	0	2		1	SIS #3	3	
12	**Lecture #12. UML diagrams and Software Documentation** Use Case, Sequence , Class diagrams. Realization, Dependency, Aggregation, Composition. Acquaintance with TopCoder UML tool. Software Documentation. Javadoc.	1	0	2	[1] Ch. 23 [2] Ch. 16	1	SIS #4	3	Checkpoint

	Coupling and cohesion.								
13	**Lecture #13. Network programming** Establishing a Simple Server Using Stream Sockets. Establishing a Simple Client Using Stream Sockets. Client/Server Interaction with Stream Socket Connections. Connecting with a URL .	1	0	2	Slides	1	SIS #4	3	
14	**Lecture #14. Design Patterns. Overview.** Singleton pattern, abstract factory pattern, prototype pattern, Decorator, etc.	1	0	2	Slides	1	Project	3	Project Defense
15	**Lecture #15 Recent Advances in Component Software – Does Scala beat Java.** About possible fusion of object-oriented and functional programming. Twitter example.	1	0	2	Slides All chapters	1		3	
	Final Exam					**In written form**			
	Total	**0**	**30**			**15**		**45**	

COURSE ASSESSMENT PARAMETERS

Type of activity	Final scores
Laboratory works	14%
Quizzes	4%
SIS	20%
Project	10%
Midterm exam	12%
Final exam	40%
Total	**100%**

Criteria for evaluation of students during semester:

	Assessment criteria	Weeks															Total scores
		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16-17

1.	Laboratory works		*		*		*		*		*		*			10%
2.	SIS			*			*		*			*				30%
3.	Project												*			10%
4.	Midterm exam							*								10%
5.	Final exam														*	40%
	Total															**100%**

Academic Policy

KBTU standard academic policy is used.

- Cheating, duplication, falsification of data, plagiarism, and crib are not permitted under any circumstances!
- Attendance is mandatory.

  Attention. Missing 20% attendance to lessons, student will be taken from discipline with filling in F (Fail) grade.

  Students must participate fully in every class. While attendance is crucial, merely being in class does not constitute "participation". Participation means reading the assigned materials, coming to class prepared to ask questions and engage in discussion.
- Students are expected to take an active role in learning.
- Written assignments (independent work) must be typewritten or written legibly and be handed in time specified. Late papers are not accepted!
- Students must arrive to class on time.
- Students are to take responsibility for making up any work missed.
- Make up tests in case of absence will not normally be allowed.
- Mobile phones must always be switched off in class.
- Students should always be appropriately dressed (in a formal/semi-formal style).
- Students should always show tolerance, consideration and mutual support towards other students.