Nirma University
Institute of Technology

Computer Science & Engineering Department

Course Policy Document

B.Tech. in Computer Science & Engineering

Semester: III: Academic Year: 2019-20, Term: Odd

Course Code & Name	:	2CS302 Object Oriented Programming
Credit Details	:	4 (L:2 T:0 P:4)
Course Co-ordinator	:	Prof. Daiwat Vyas
Contact No. & Email id	:	9662276830 daiwat.vyas@nirmuni.ac.in
Office	:	D306/2 Staff Room
Visiting Hours	:	9 AM to 4 PM
Course Website Link:	:	https://sites.google.com/a/nirmauni.ac.in/2cs302-oop-odd-sem-2019-20/
Course Faculties:	:	Prof. Daiwat Vyas (D306/2) (daiwat.vyas@niramuni.ac.in) Prof. Ajaykumar Patel (D306/4) (ajaypatel@niramuni.ac.in)
Email id	:	Prof. Anuja Nair (N- Block 5th Floor Faculty Cabin) (anuja.nair@niramuni.ac.in) Prof. Kavita Tewani (N- Block 5th Floor Faculty Cabin) (kavita.tewani@nirmauni.ac.in
Office	:	
Visiting Hours	:	9 AM to 4 PM (Mon to Fri and 1st and 3rd Sat)

1. Introduction to Course

1.1 Importance of the course

- Object-oriented programming provides a superior way of organizing programming projects:
  - It encourages a high degree of modularity in programming, making projects easier to implement.
  - It provides powerful techniques like inheritance and polymorphism to help organize and reuse code.
- Object-oriented languages like Java provide a good environment for beginners to learn programming.

1.2 Objective of the Course

Main objective of this course is to make the students understand basic principles and constructs of object oriented programming. Make them aware about the tools and technologies used for modular development and also make them able to design, develop, execute and validate program in object oriented programming environment.

In this course student will not only learn all the core Object Oriented Programming concepts, but also learn how and when to transfer those concepts into programming language such as Java.

1.3 Pre-requisite:

Basic understanding of programming required.

2. Course Learning Outcomes

After successful completion of the course, a student will be able to:

1. interpret the basic principles of object oriented programming.
2. develop computer programs to solve real world problems based on object-oriented principles.
3. implement multi-threaded applications with basic input-output operations and exception handling.

3. Syllabus

Syllabus:	Teaching Hours:
Unit I **Introduction:** A Review of programming paradigms, Introduction to Object Oriented Programming, Comparison of Object Oriented approach with other programming approaches	**02**
Unit II **History and Overview of Java**: Creation of Java, , Evolution of Java, features of Java, byte code, Java Development Kit (JDK), Java Virtual Machine (JVM) , Introduction to three OOP principles (Inheritance, Polymorphism, Encapsulation), Introduction to classes and methods	**02**
Unit III **Data Types, Variables and Operators in Java:** Arrays: One dimensional array, multi-dimensional array, alternative array declaration statements. Control Statements like Selection statements (i.e if, switch etc.), iteration statements (i.e while, do-while, the for-each version of the for Loop, Nested Loops etc.) , jump statements (i.e break, continue)	**06**
Unit IV **Classes and Methods:** class fundamentals, objects, assigning object reference variables, methods in class, constructors, this keyword, garbage collection, finalize() method, overloading methods, argument passing, access control, static, final, nested and inner classes, command line arguments, variable-length arguments. String Handling: Basics of String handling in Java, String class methods, String Buffer Class methods Inheritances: Basics, member access and inheritance, super class references, using super, multilevel hierarchy, constructor call sequence, method overriding, dynamic method dispatch, abstract classes, Object class Packages and Interfaces: defining a package, finding packages and CLASSPATH, access protection, importing packages, interfaces (defining, implementation, nesting, applying), variables in interfaces, extending interfaces, instance of operator	**10**
Unit V **Exception Handling**: fundamental, exception types, uncaught exceptions, try, catch, throw, throws, finally, multiple catch clauses, nested try statements, built-in exceptions, custom exceptions (creating your own exception sub classes). Multithreaded Programming: Java thread model, thread priorities, synchronization, messaging, Thread class, Runnable interfaces, creating a thread(s), Thread class methods, Synchronization, Inter thread Communication, volatile operators. Managing I/O: Streams, Byte Streams and Character Streams, Predefined Streams, Reading	**10**

console Input, Writing Console Output, PrintWriter class, File management classes	

3.1. Self-Study

The self-study components will be as mentioned below and around 10% of the questions will be asked from self-study content.

Topics/content for self-study are as listed below:

Managing I/O: Streams, Byte Streams and Character Streams, Predefined Streams, Reading console Input, Writing Console Output, PrintWriter class, File management classes.

3.2. Reference Books:

1. Herbert Schildt, Java – The Complete Reference, Tata McGraw Hill

2. Balaguruswamy, Programming with Java – A primer, Tata McGraw Hill

3. David Flanagan, Student Workbook Java in a Nutshell O'Reilly

4. Cay S. Horstmann Core Java(TM), Volume I—Fundamentals Prentice Hall

Note: The latest edition of books should be referred.

4. Laboratory details

Laboratory work will be based on above syllabus with following experiments to be performed.

Each experiment will be of 10 marks. Evaluation for 100 marks will be done throughout the semester as part of the Continuous Evaluation scheme. The assessment of Laboratory work is as under:

Total Marks	**Continuous Evaluation**			**Semester End Evaluation**	
100 marks	**No. of Practicals**	**Max. Marks**	**Weightage**	**Max Marks**	**Weightage**
	10	100	75%	25	25%

Sr. No.	Name of Practical	Hour(s)	Max. Marks	CLO	Week
0	Setting up Java Programming Environment and exploration of programming paradigms.	02	--	--	1
0	Hands-on practice on C- Programming	02	--	--	1
	Take all inputs (wherever required) from user after practical no. 1 onwards				
1	**(a)** Write a Java program to display greeting message like: "Hello! Java" on console. **(b)** Write a Java program to display all primitive type variables. Also display your name in the last line. **(c)** Justify the following statement in the context of Java. " boolean can be true (Non-zero) or false(Zero)".	02	10	1,2	2
2	**(a)** Write a Java program to print the ASCII values for characters entered by the user. ASCII value of A = 65 ASCII value of B = 66.....and So on. Print all the ASCII values on screen. **Hint:** First 32 characters are non-printing, where 32 itself is an ASCII code for space. Hence cover the characters from 32 and onwards. **(b)** Write a Java program to get particulars of his/her birthday and display it as shown below. Use 3 variables to hold date, month and year. Your birthday is 24/07/1989	04	10	1,2	2,3
3	**(a)** Write a Java program using class that prints the numbers 1 to 50. For all multiples of 3 print "Fizz" and for all multiples of 5 print "Bizz". For multiples of both 3 and 5 print "Fizz-Bizz". **(b)** Demonstrate concept of Arithmetic & Bitwise Operators with a java program. Operands to be considered as per the operators entered by the user. **(c)** Write a java program which generates student grade report in console. Take student roll number and marks (out of 100) of 5 courses from user. Calculate the percentage and display grade of the student. Use appropriate control statements. **(d)** Write a program to calculate area and perimeter of a circle. Take the value of radius from user.	04	10	1,2	3,4
4	**(a)** Write a Java program to decide the following information based on Body Mass Index. Let the user enter height in feet and inch and weight in pounds (lb). (Hint: 1 feet = 12 inches). Based on BMI	04	10	1,2	4,5

	computed, print relevant message i.e if BMI is <18.5 print "Person is Under-weight", if BMI is >18.5 & < 24.9 print "Person is having Normal BMI" & if BMI is >25 & <29.9 print "Person is Over-weight", if BMI>30 print "Person Is Obese". **(b)** Write a Java program to find all even numbers between1 and a given number given as input by user. **(c)** Write a Java program to: **i.** check whether a number is odd or even (using if – else statement) **ii.** check the category of a given character. (using if…else…if ladder) **iii.** check whether a number is prime or not. (using for loop) **iv.** display reverse of a number and check whether it is palindrome or not. (using while/do while loop) **v.** perform arithmetic operations of a calculator. (using switch case) **vi.** pattern printing. (using nested loops) 1 1 2 1 2 3 1 2 3 4 1 2 3 4 5 1 2 3 4 5 6				
5	Write a Java program to **(a)** that allows you to create an integer array of 18 elements with the following values: int A[ ] = {3, 2, 4, 5, 6, 4, 5, 7, 3, 2, 3, 4, 7, 1, 2, 0, 0, 0}. The program computes the sum of element 0 to 14 and stores it at element 15, computes the average and stores it at element 16 and identifies the smallest value from the array and stores it at element 17. **(b)** sort given n numbers and display them in ascending and descending order. **(c)** to add two given matrices. **(d)** to multiply two given matrices.	02	10	1,2	5
6	**(a)** Define a class Rectangle with its length and breadth. Provide appropriate constructor(s), which gives facility of constructing rectangle object with default values of length and breadth as 0 or passing value of length and breadth externally to constructor.	08	10	1,2	6,7

Provide appropriate accessor & mutator methods to Rectangle class.

Provide methods to calculate area & to display all information of Rectangle.

Design different class TestRectangle class in separate source file, which will contain main function. From this main function, create 5 Rectangle objects by taking all necessary information from the user.

The class has attributes length and width, each of which defaults to 1. It should have member functions that calculate the perimeter and area of the rectangle. It should have set and get functions for both length and width. The set functions should verify that length and width are each floating-point numbers larger than 0.0 and less than 20.0.

(b) Create a class Term. This class represents a term of a polynomial such as 2x4 where 2 is coefficient and 4 is exponent of the term.
Data members:-

- int coefficient
- int exponent

Create another class Polynomial. The internal representation of a polynomial is an array of Terms. The size of this array should be fixed.
Provide a constructor for this class that will set all terms of a polynomial object as zero (where coefficient is 0 and exponent is 0). Provide following functions:

- setTerm(int, int) – Setting a term of a polynomial object. Each successive call of this function should set next term of the polynomial object.

It should do the following validations:-

- Whether the exponent of the term being set is already used.
- Whether the array size limit is exceeded.
- Whether the exponent is negative.

In all the cases it should not set the term and display an appropriate message.

- sort() – to arrange the terms in ascending order of exponents.
- provide a function to print a polynomial object

(c) Create a class called complex for performing arithmetic operations with complex numbers. Use floating point variables to represent the private data of

	the class. Provide a default constructor that initializes the object with some default values. Provide public member functions for each of the following • Addition of two complex numbers: It returns the result obtained by adding the respective real parts and the imaginary parts of the two complex numbers. • Subtraction of two complex numbers: It returns the result obtained by subtracting the respective real parts and the imaginary parts of the two complex numbers. • display( ) – It displays the complex number in a+bi format. The output should be displayed as follows:- Sum of a1+b1 i & a2+b2 i is : a3+b3 i **(d)** Create an object called GSSArray. (It stands for growable self-sorting array) This object will manage an array of type int. Create a private variable for an array of type int. In the constructor for this object, take in an int value which will determine the starting size of the array. The constructor should also instantiate the array. Create a public method called insert, which will take in an int and find the location in the array where it belongs and insert it there. If the array is full, then before inserting the value, method insert should call private method increaseSize, which will create a new array which is double the size of the current array. Then it will copy the values from the original array into the new array and set the private variable to this new array. The array should keep track of how many of its indexes are filled. Create a private variable called lastindex which will be equal to the last index of the array that has a value. Create a public method delete, which will take an int and if will remove the 1st instance of that number in the array. If the number doesn't exist, the method should return false, otherwise it should return true. (Don't forget to update variable lastindex in methods delete and insert.)				
7	**(a)** Write a program to perform following operations on string “Nirma University” • Reverse the string • Replace character Ni with Ab • Check whether strings “rma” and “Uni” present in original string or not • Compare this program implementation using String and StringBuffer methods.	06	10	1,2	7,8

	(b) Write a program to reverse words in a string. For example, if input is "Welcome to Nirma". Output should be "emocleW ot amriN". **(c)** Write a program to find longest substring which is a palindrome. For example, if input is "fornirmaamrinfor", output should be "nirmaamrin". **(d)** Accept a paragraph of text consisting of sentences that are terminated by either '.' (full stop), '!' (exclamation mark) or a '?' (question mark). Assume that there can be maximum 10 sentences in a paragraph. Write a program to arrange the sentences in increasing order of their number of words. For Example : INPUT: Please come and attend the party. Hello! How are you? OUTPUT : Hello = 1 How are you = 3 Please come and attend the party = 6				
8	**(a)** Create an abstract class Instrument which is having the abstract function play. Create three more sub classes from Instrument which is Piano, Flute and Guitar. Override the play method inside all three classes printing a message. "Piano is playing tan tan tan tan" for Piano class "Flute is playing toot toot toot toot" for Flute class "Guitar is playing tin tin tin" for Guitar class You must not allow the user to declare an object of Instrument class. Create an array of 10 Instruments. Assign different type of instrument to Instrument reference. Check for the polymorphic behavior of play method. Use the instanceof operator to print that which object stored at which index of instrument array. **(b)** Create an abstract class Compartment to represent a rail coach. Provide an abstract function notice in this class. Derive FirstClass, Ladies, General and Luggage classes from the compartment class. Override the notice function in each of them to print notice suitable to the type of the compartment. Create a class TestCompartment. Write main function to do the following: Declare an array of Compartment pointers of size 10.	08	10	1,2	8,9

	Create a compartment of a type as decided by a randomly generated integer in the range 1 to 4. Check the polymorphic behavior of the notice method. **(c)** Create a class Medicine to represent a drug manufactured by a pharmaceutical company. Provide a function displayLabel( ) in this class to print Name and address of the company. Derive Tablet, Syrup and Ointment classes from the Medicine class. Override the displayLabel( ) function in each of these classes to print additional information suitable to the type of medicine. For example, in case of tablets, it could be "store in a cool dry place", in case of ointments it could be "for external use only" etc. Create a class TestMedicine. Write main function to do the following: Declare an array of Medicine references of size 10 Create a medicine object of the type as decided by a randomly generated integer in the range 1 to 3. Check the polymorphic behavior of the displayLabel( ) method. **(d)** Create a class Car which contains members speed, noOfGear. The class has a method drive( ) which is responsible to provide starting speed and noOfGears to a Car. Implement display( ) method which will display all attributes of Car class. The class SportCar is derived from the class Car which adds new features AirBallonType. When this method is invoked, initial speed and gear status must be displayed on console. Override the display method which display all attribute of the SportCar. Make use of super class display( ) method. **(e)** A super class Record has been defined to store the names and ranks of 50 students. Define a sub class Rank to find the highest rank along with the name. The details of both classes are given below: Class name : Record Data Members / instance variables: • name[ ] : to store the names of students • rnk[ ] : to store the ranks of students Member functions: • Record( ) : constructor to initialize data members • void readvalues( ) : to store names and ranks • void display( ) : displays the names and the corresponding ranks Class name : Rank Data Members / instance variables:				

	• index : integer to store the index of the topmost rank Member functions • Rank( ) : constructor to invoke the base class constructor and to initialize index to 0. • void highest( ) : finds the index location of the topmost rank and stores it in index without sorting the array 6 • void display( ) : displays the name and ranks along with the name having the topmost rank. Specify the class Record giving details of the constructor( ), void readvalues( ), void display( ). Using the concept of inheritance, specify the class Rank giving details of constructor( ), void highest( ) and void display( ).				

9	**a)** Create a package nirma.st.itnu<your batch id>.cs<your roll number>.<your first name>.
	For example, nirma.st.itnu18.cs270.ABC
	Now create a Greeter class in this package having the following features:
	Attributes:
	• name string //indicates name of the person to be greeted
	Member functions:
	• Greeter(aName) //constructor to initialize the name of the //person to be greeted by this greeter.
	• sayHello( ) //returns a hello message with the name of the //person initialized earlier.
	• sayGoodBye( ) //bids goodbye to the person named earlier.
	Create another class in the same package called Advisor that has the following features:
	Attributes:
	• message string[5] //contains five advice messages
	Member functions:
	• Advisor( ) //default constructor to initialize an array of //strings with atleast five advice messages
	• getAdvice( ) //randomly selects an advice from the available //list of messages and returns it to the caller of //this method
	Outside the package, from your working directory, create a class GreeterTest that constructs Greeter objects for all command-line arguments and prints out the results of calling sayHello( ).
	The program should then display an advice and finally bid goodbye to each of the persons/entities in reverse order of the names entered at the command line.
	For example,
		10	10	1,2	10,11 12

java GreeterTest Mars Venus
then the program should print

Hello, Mars!
Hello, Venus!
Advice: Never say No
Goodbye Venus!
Goodbye Mars!

(b) Create a class called CalcAverage that has the following method:
public double avgFirstN(int N)
This method receives an integer as a parameter and calculates the average of first N natural numbers. If N is not a natural number, throw an exception IllegalArgumentException with an appropriate message.

(c) Create a class Number having the following features:
Attributes:

int	first number
int	second number
result	double

stores the result of arithmetic operations performed on a and b
Member functions:

- Number(x, y) : constructor to initialize the values of a and b
- add( ) : stores the sum of a and b in result
- sub( ) : stores difference of a and b in result
- mul( ) : stores product in result
- div( ) : stores a divided by b in result

Test to see if b is 0 and throw an appropriate exception since division by zero is undefined.
Display a menu to the user to perform the above four arithmetic operations.

(d) Create a class BankAccount having the members as given below:
accNointeger
custName string
accType string (indicates 'Savings' or 'Current')
balance float
Include the following methods in the BankAccount class:

- void deposit(float amt);
- void withdraw(float amt);
- float getBalance();

deposit(float amt) method allows you to credit an amount into the current balance. If amount is negative, throw an exception NegativeAmount to block the operation from being performed.

withdraw(float amt) method allows you to debit an amount from the current balance. Please ensure a minimum balance of Rs. 1000/- in the account for savings account and Rs. 5000/- for current account, else throw an exception InsufficientFunds and block the withdrawal operation. Also throw an exception NegativeAmount to block the operation from being performed if the amt parameter passed to this function is negative.
getBalance( ) method returns the current balance. If the current balance is below the minimum required balance, then throw an exception LowBalanceException accordingly.
Have constructor to which you will pass, accno, cust_name, acctype and initial balance.
And check whether the balance is less than 1000 or not in case of savings account and less than 5000 in case of a current account. If so, then raise a LowBalanceException.
In either case if the balance is negative then raise the NegativeAmount exception accordingly.

(e) Create a class with following specifications.
Class Emp

empId	int	
empName	string	
designation	string	
basic	double	
hra	double	readOnly

Methods

- printDET( )
- kalculateHRA( )
- printDET( ) methods will show details of the EMP.

calculateHRA( ) method will calculate HRA based on basic.
There will 3 designations supported by the application.

If designation is “Manager” - HRA will be 10% of BASIC
if designation is “Officer” - HRA will be 12% of BASIC
if category is “CLERK” - HRA will be 5% of BASIC
Have constructor to which you will pass, empId, designation, basic and price.
And checks whether the BASIC is less than 500 or not. If it is less than 500 raise a custom Exception as given below
Create LowSalException class with proper user message to handle BASIC less than 500.

	(e) Create a class USERTRAIL with following specifications. val1, val2 type int Method boolean show( ) will check if val1 and val2 are greater or less than Zero Have constructor which will val1, val2 and check whether if it is less than 0 then raise a custom Exception (name: Illegal value exception.)				
10	**(a)** Implement three classes: Storage, Counter and Printer. The Storage class should store an integer. The Counter class should create a thread and starts counting from 0 (0, 1, 2, 3…) and stores each value in the Storage class. The Printer Class should create a thread that keeps reading the value in the Storage class and printing it. Write a program that creates an instance of the Storage class and set up a Counter and Printer object to operate on it. Identify that, whether synchronization is required or not in this assignment. If yes, implement it. **(b)** Modify the above program in assignment no. 1 to ensure that each number is printed exactly once, by adding suitable synchronization. **(c)** Write a multithreaded program that will accept 4 strings from the command line and search in a particular file for a given string and display the status of each search on the screen. Note that, all threads are operating on the same file. **(d)** Write a Java application that will accept two filenames (text files) as command line arguments and use two threads to read contents from the two text files. Each of the threads should sleep for a random time after displaying filename with each line. **(e)** Write a java application that will create and start two threads. One thread will read a text file (Number.txt) containing five positive integers one on each line. The second thread should calculate factorial of the number read by the first thread and print the message on the screen as “Factorial of x is y”, here x is number & y is factorial of the number The two threads should work in synchronization. Handle all necessary exceptions.	12	10	1,2	13,14, 15

	(f) Write a stream based program which will accept Roll Number, Name, Age and Address from user. Age and Roll-no should be numeric. Handle with built-in exception. None of the field should be blank. Handle with custom exception. Ask user, whether to write the data in the file. If answer is yes, then data is saved into a file as an object (User can write many records in the file), otherwise terminate the current program. Write another program to display all the records saved into the file				
	Total Hours for Practicals 1 to 10	60 Hours			
	*** Extra Practicals for Practice (Not to be considered for evaluation purpose)**				
11*	Create a package nirma.itnu.shape containing the following classes and interfaces. An interface Polygon containing the members as given below: area float perimeter float void calcArea( ); abstract method to calculate area of a particular polygon given its dimensions void calcPeri( ); abstract method to calculate perimeter of a particular polygon given its dimensions void display( ); method to display the area and perimeter of the given polygon Create a class Square that implements Polygon and has the following member: side float Square(float s); constructor to initialize side of square Create another class Rectangle that implements Polygon and has the following member: length float breadth float Rectangle(int len, int bre); constructor to initialize length and breadth of a rectangle Outside the package, create a class that imports the above package an instantiates an object of the Square class and an object of the Rectangle class. Call the above methods on each of the classes to calculate the area and perimeter given the side and the length/breadth of the Square class and the Rectangle class respectively.	--	--	1,2	--

12*	**Write a java program to implement the concept of user defined exception for Employee Management System. All the details should be stored in a text file and your program should provide an option to retrieve the data based on user's request and display it on console window. Menu options should be given to user.** **Decide where user-defined exception is required.** **Sample Output:** Enter employee-1 details: Enter employee no and name: -5554 Exception caught: enter valid number Enter employee-1 details: Enter employee no and name: 3001 Raj Dave Enter job and salary: Lecturer 25000 Enter employee-2 details: Enter employee no and name: 3002 Kinjal Dave Enter job and salary: Asst. Professor 40000 The final output should be stored in a text file and once user selects an option to display the details on console it should retrieve and display content from text file as given below: **ENO ENAME JOB SALARY** 3001 Raj Dave Lecturer 25000 3002 Kinjal Dave Asst. Professor 40000	--	--	3	--
13*	**Implement a java program for scenario as given below:** Encryption is a technique of coding messages to maintain their secrecy. A String array of size 'n' where 'n' is greater than 1 and less than 10, stores single sentences (each sentence ends with a full stop) in each row of the array. Write a java program to accept the size of the array. Display an appropriate message if the size is not satisfying the given condition. Define a string array of the inputted size and fill it with sentences row-wise. Change the sentence of the odd rows with an encryption of two characters ahead of the original character.	--	--	1,2	--

	Also change the sentence of the even rows by storing the sentence in reverse order. Display the encrypted sentences as per the sample data given below. Test your program on the sample data and some random data. Sample Input and Output: Input: n=4 IT IS CLOUDY. IT MAY RAIN. THE WEATHER IS FINE. IT IS COOL. Output: KV KU ENQWFA. RAIN MAY IT. VJG YGCVJGT KU HKPG. COOL IS IT. Input: n=13 Output: INVALID ENTRY				
14*	**Implement a java program for scenario as given below:** Write a program which takes a string (maximum 80 characters terminated by a full stop. The words in this string are assumed to be separated by one or more blanks. Arrange the words of the input string in descending order of their lengths. Same length words should be sorted alphabetically. Each word must start with an uppercase letter and the sentence should be terminated by a full stop. **In the end store the final output in a text file.** Test your program for the following data and some random data. SAMPLE DATA: INPUT: "This is human resource department." OUTPUT: Department Resource Human This Is. INPUT: "To handle yourself use your head and to handle others use your heart."	--	--	1,2	--

	OUTPUT: Yourself Handle Handle Others Heart Head Your Your And Use Use To To.				

#- For Evaluation only 1 to 10 Practicals will be considered, 11, 12, 13 are extra practicals for practice.

Schedule

Students will have two labs per week for this course in their semester.

Assessment policy for Lab Practicals

Maximum marks for each practical is 10.
Each practical will be assessed by the lab faculty. Student's lab performance will be evaluated based on following parameters:

- Code execution
- Logic
- Syntax
- Error Solving
- Viva
- Timely submission of practical's and lab write up file
- Regularity in attending lab sessions
- Discipline of student in the lab sessions

Lab Practical File Submission

OOP Lab Practicals have to be submitted hand-written in following format in file pages:

Date:

Roll No. and Name:

Course Code and Name:

Practical No.

AIM:

Methodology followed:

Theoretical Principles used: explain them in your own way by explaining theory and using small examples

Input:

Output:

Conclusion:

Signature of Teacher:

6. Course Evaluation Methodology

6.1 Component wise Continuous Evaluation (CE), Laboratory and Project Work (LPW) & Semester End Examination (SEE) weightage

Assessment scheme	**CE**			**LPW**		**SEE**
Component weightage	0.4			0.2		0.4
	Class Test 30%	Sessional Exam 40%	One Conceptual Test 30%	Continuous Evaluation 75%	Viva Voce 25%	

6.2 Assessment Policy for Continuous Evaluation (CE)

Assessment of Continuous Evaluation comprises of three components:

1. Class Test will be conducted as per academic calendar. It will be conducted online/ offline for the duration of 1 hour and will be of 30 marks.
2. Sessional Exam will be conducted as per academic calendar. It will be conducted offline for the duration of 1 hour and 15 minutes and will be of 40 marks.
3. There will be one Conceptual Test (CT) conducted after sessional exam and before final LPW exam, the exact date of exam will be conveyed by the subject faculty. It will be conducted online for the duration of 1 hour and will be of 30 marks.

6.3 Assessment Policy for Laboratory (LPW)

Assessment of Laboratory and Project Work comprises of two components.

1. Continuous assessment for laboratory experiments will be conducted. There will be 10 experiments, each carrying weightage of 10 marks. At the end of the course total marks obtained out of 100 will be converted according to weightage assigned. Assessment of Experiment will be carried out based on parameters like Completion of lab work file, understanding of the experiment performed, originality, involvement of the student, regularity, discipline etc. during the session and the parameters mentioned in Section 4 of this document.

2. A Viva voce examination for LPW component will be conducted as per academic calendar. It will carry a weightage of 25 marks.

6.4 Assessment Policy for Semester End Examination (SEE)

A written examination of 3 hour duration will be conducted for the course as per academic calendar. It will carry 100 marks and marks obtained out of 100 will be converted as per weightage assigned.

7. Lesson Plan

Lect. No.	Topic	Mapped CLO
1	**Introduction to Course and Course Policy presentation** A Review of programming paradigms, Introduction to Object Oriented Programming	1
2	Comparison of Object Oriented approach with other programming approaches	1
3	**History and Overview of Java:** Creation of Java, , Evolution of Java, features of Java, byte code, Java Development Kit (JDK), Java Virtual Machine (JVM)	1
4	Introduction to three OOP principles (Inheritance, Polymorphism, Encapsulation), Introduction to classes and methods	1,2
5	**Data types, variables and Operators**: primitive data types, literals, variables, type conversion and casting	1,2
6	Automatic type promotion in expressions, type promotion rules, Basic of Array and String	1,2
7	Arithmetic operators, bitwise operators	1,2
8	relational operators, Boolean logical operators, Assignment operators, ternary operators, operator precedence	1,2
9	**Control Statements**: Selection statements : if, switch **Iteration statements :** while, do-while, for,	1,2
10	**Iteration statements :**the for-each version of the for Loop, Nested Loops **Jump statements:** break and continue **Arrays**: one dimensional array, multi-dimensional array, alternative array declaration statements	1,2

11	**Classes and Methods**: class fundamentals, declaring objects, assigning object reference variables **Introduction to method:** Adding a method to the class, returning a value, Adding a Method that takes parameters	1,2
12	Constructors, Parameterized constructor, Garbage collection, Overloading methods, overloading constructors	1,2
13	this keyword , finalize() method, Object as a parameter, Returning objects, Recursion	1,2
14	Access control: static, final, Nested and inner classes, Command line arguments, variable-length arguments	1,2
15	**String Handling:** String constructors, String Operations	1,2
16	String Methods, String Buffer Class	1,2
17	**Inheritances**: Basics, member access and inheritance, Super class references, Using super	1,2
18	Multilevel hierarchy, constructor call sequence, method overriding, Dynamic method dispatch, abstract classes, Object class	1,2
19	**Packages and Interfaces**: defining a package, finding packages and CLASSPATH, access protection	1,2
20	Importing packages, interfaces (defining, implementation, nesting, applying), variables in interfaces, extending interfaces, instance of operator	1,2
21	**Exception Handling**: fundamental, exception types, uncaught exceptions, try, catch,	3
22	throw, throws, finally	3
23	multiple catch clauses, nested try statements	3
24	built-in exceptions, custom exceptions (creating your own exception sub classes)	3
25	**Multithreaded Programming**: Basics of Multithreading, Java thread model	3
26	thread priorities, synchronization, messaging	3
27	Thread class, Runnable interfaces	3
28	creating a thread(s), Thread class methods	3
29	Synchronization, Inter thread Communication	3

30	**Managing I/O**: Streams, Byte Streams and Character Streams, Predefined Streams, Reading console Input, Writing Console Output PrintWriter class, File management class (Self Study)	3

Note: Class Test will be conducted in 6th week and Sessional Exam in 12th week. Conceptual Test will be conducted after sessional exam and before final LPW exam, the exact date of exam will be conveyed by the subject faculty.

8. Mapping of Session Learning Outcomes (SLO) with Course Learning Outcomes (CLO)

Session No.	Session Learning Outcomes: After successful completion of the session, student will be able to:	CLO
0.	understand the importance and scope of the course and it's application areas	--
1.	understand the need of introducing a new programming language, Java	1
2.	understand and compare Object Oriented approach with other programming approaches	1
3.	Understand features of Java, byte code, Java Development Kit (JDK), Java Virtual Machine (JVM)	1
4.	understand the basic principles of OOP and basics of Classes in Java	1,2
5.	▪ know how to use data types, variables in Java ▪ understand concepts of type casting and type conversion and apply those in programming	1,2
6.	understand rules of Automatic type promotion in expressions and apply those in programming	1,2
7.	know how to use different operators in Java	1,2
8.	know how to use different operators in Java	1,2
9.	understand concepts of control statements & iteration statements in Java and apply those in programming	1,2
10.	▪ understand concepts of jump statements in Java and apply those in programming ▪ know how to apply concepts of arrays in Java programming	1,2
11.	▪ understand basic concepts of class fundamentals apply concepts of classes in programming ▪ understand concepts of methods in java and apply those in programming	1,2
12.	understand and apply concepts of Overloading Methods, Overloading Constructors in Java	1,2
13.	▪ understand and apply concepts of this keyword , finalize() method in Java	1,2

	▪ understand and apply concepts of Object as a parameter, Returning objects, Recursion in Java	
14.	▪ understand and apply concepts of Access control: static, final, Nested and inner classes in Java ▪ understand and apply concepts of Command line arguments, variable-length arguments in Java	1,2
15.	know concepts of **String Handling:** String constructors, String Operations and apply in programming	1,2
16.	▪ create programs using String Methods for performing string manipulation ▪ create programs using String Buffer Class for performing string manipulation	1,2
17.	understand and apply concepts of **Inheritances**: Basics, member access and inheritance, Super class references, Using super in Java	1,2
18.	▪ understand and apply concepts of Multilevel hierarchy, constructor call sequence, method overriding, in Java ▪ understand and apply concepts of Dynamic method dispatch, abstract classes, Object class in Java	1,2
19.	know how to use **Packages and Interfaces**: defining a package, finding packages and CLASSPATH, access protection, in Java	1,2
20.	know concepts about Importing packages, interfaces (defining, implementation, nesting, applying), variables in interfaces, extending interfaces, instance of operator in Java	1,2
21.	understand and apply concepts of **Exception Handling**: fundamental, exception types, uncaught exceptions, try, catch, in Java	3
22.	understand and apply concepts of throw, throws, finally, in Java	3
23.	create built-in exceptions, custom exceptions (creating your own exception sub classes) in Java	3
24.	understand and apply concepts of multiple catch clauses, nested try statements in Java	3
25.	understand and apply concepts **Multithreaded Programming**: Basics of Multithreading, Java thread model	3
26.	understand and apply concepts of **Multithreaded Programming**: thread priorities, synchronization, messaging	3
27.	understand and apply concepts of Thread class, Runnable interfaces in Java	3
28.	apply concepts of creating a thread(s), Thread class methods in Java	3
29.	apply concepts of Synchronization, Inter thread Communication	3
30.	▪ understand and apply concepts of **Managing I/O**: Streams, Byte Streams and Character Streams, Predefined Streams in Java	3

	▪ know how to Reading console Input, Writing Console Output in Java File Handling ▪ know how to use PrintWriter class, File management class in Java	

9. Teaching-learning methodology

1. **Lectures:** Primarily Chalk and Black board will be used to conduct the course. However, where required, Power Point Presentations (PPTs), Video Lectures, Simulations / Animations etc. will be used to enhance the teaching-learning process.
2. **Laboratory:** Explanation of Experiment to be performed along with co-relation with theory will be given. At the end of each session assessment will be carried out based on parameters like completion of lab work that includes observations, calculations, graphs and conclusions, individuality and involvement of the student, regularity, discipline etc. Students will be quizzed to check their understanding of the experiment/exercise conducted.

10. Active learning techniques

Following active learning techniques will be adopted for the course.

- Think-Pair-Share: Think-Pair-Share (TPS) is a collaborative learning strategy in which students work together to solve a given problem and/or answer a question based on given learning materials. In class Programming exercises and concept discussion for different OOP concepts.
- Flipped Class-room (Arrays, Control Statements)

11. Course Material

Following course materials will be uploaded on the course website:

- Course Policy
- Lecture Notes (will be provided just before exam & not day to day)
- Books / Reference Books / NPTEL video lectures
- Assignments, Tutorials, Lab Manuals
- Question bank (As and when topics are covered in class and lab sessions)
- Web-links, Video Lectures for reference
- Animations /Simulations, Softwares
- Advanced topics

12. Course Learning Outcome Attainment

Following means will be used to assess attainment of course learning outcomes:

- Use of formal evaluation components of continuous evaluation, tutorials, laboratory work, semester end examination
- Informal feedback during course conduction

13. Academic Integrity Statement

Students are expected to carry out assigned work under Continuous Evaluation (CE) component and LPW component independently. Copying in any form is not acceptable and will invite strict disciplinary action. Evaluation of corresponding component will be affected proportionately in such cases. Turnitin software will be used to check plagiarism wherever applicable. Academic integrity is expected from students in all components of course assessment.