Online Diploma in Advanced Computing (e-DAC)

May 2021 Batch

EVALUATION GUIDELINES

1. Evaluation

Evaluation is an essential part of conducting the online Diploma in Advanced Computing (e-DAC), as it provides important feedback and inputs to both the centre and the students. The centre gets an idea about the relative performance of each student, which also serves as feedback about the design and conduct of the course. The student gets a clear picture of his/her academic standing, individually and in comparison to his/her fellow students.

In order to ensure timely and efficient evaluation and certification of all students, the following guidelines are being issued and should be followed religiously.

2. Evaluation Methodology

- Each centre should have a Designated Responsible Member (DRM) for evaluation.
- The DRM Evaluation would be responsible for coordinating all activities relating to evaluation at the training centre and for communicating with the nodal CDAC centre.
- Evaluation is a compulsory part of the process of obtaining the DAC certificate. All students are required to pass each module of the course in order to be eligible to receive the Diploma Certificate.
- The faculty of every module should outline the objectives of the evaluation to be conducted for that module, so as to enable the students to prepare themselves properly.

3. Modular Evaluation

3.1 Subject-wise Evaluation

a. A separate evaluation process is to be conducted for every module of the course. The evaluation for each module must be completed as per the guidelines given below. The mid-module/surprise test evaluation is mandatory and can be taken after discussion with the concerned faculty.
b. Students are evaluated on a continuous basis throughout the duration of the course. To have a very uniform and fair assessment, the evaluation process is divided into two parts:

(1) Continuous Assessment – CA (60marks)
(2) Course End Examination – CE (40 marks)

3.2 Continuous Assessment

This is being done primarily by the respective faculty in the form of Lab tests, assignments, quizzes etc conducted with the help of the respective course coordinators at regular intervals, as and when the portions of the modules are completed. These are basically internal exams and local to the centre. This process is further categorized into two parts.

(i) Lab test (40 marks)
(ii) Internal test (20 marks): Assignments, case studies, quiz, viva, group discussions, etc. depending on the subject and the faculty.

3.3 Weightage of Marks

The figures shown below indicate the weightage of each component of a module in the final performance statement. The examination(s) for each module must be conducted for at least that number of marks. However, the centre may conduct evaluation for a higher number of marks, in which case the marks will be scaled down. For example, if the lab test for a module is conducted for 100 marks, the marks earned by the students will be scaled down to out of 40.

The weightage for each component will normally be:

Theory	- 40% (Through Centralized Course End Examinations, ie. CCEE)
Laboratory	- 40% (Lab part of Internal Assessment)
Internal Test	- 20% (Internals part of Internal Assessment)

Note: Where a module does not have a practical component, the lab component weightage will be merged with the Internal Test weightage.

3.4 Passing a Module

A student must score a minimum of 40 percent marks in each component of the evaluation, and also in the aggregate score, in order to successfully clear the module. If a student scores more than 40% on aggregate but has scored less than 40% in one component of the evaluation, he/she will not be declared as passed.

4. e-DAC Marks Distribution

The figures shown below indicate the weightage of each subject in the final performance statement for e-DAC.

Subject Code	**Module Name**	**Contact Hours**			**No of Questions in CCEE & Marks**	**Marks**			
		Theory	**Lab**	**Total**		**CCEE**	**Lab**	**IA**	**Total**
DAC01	Concepts of Programming, Operating System & Software Engineering	36	28	64	Programming Concepts: 12 Operating System: 16 Software Engineering: 12	40	40	20	100
DAC02	Object Oriented Programming with Java	44	44	88	OOP with Java: 40	40	40	20	100
DAC03	Algorithms and Data Structures(Using Java)	32	28	60	ADS: 40	40	40	20	100
DAC04	Database Technologies	30	30	60	Database: 40	40	40	20	100
DAC05	Web Programming Technologies	44	44	88	Web Programming: 40	40	40	20	100
DAC06	Web-based Java Programming	42	38	80	Web Java: 40	40	40	20	100
DAC07	Aptitude & Communication			60	***(No CCEE)*** Aptitude: 50 Communication: 50	50	0	50	Grade
DAC08	Software Project	00	00	100	Phase I: 20 Phase II: 80 (Mid Evaluation: 20 + Final Evaluation: 60)	0	0	100	Grade
	Total			**600**	**-**	**240**	**240**	**120**	**600**

5. Centralized Course-End Examinations (CCEE)

There will be Centralised Course End Examination (CCEE) after completion of all the modules of e-DAC. These exams will test the knowledge of the students about the concepts of each module and is a compulsory part of the evaluation. Conducting CCEE involves performing duty with responsibility. As a small mistake in the process may hamper the whole system, everyone has to play their role in an effective manner. It is a joint effort which has to be carried out in a combined way. Right from receiving the question paper from the national resource centres (NRC) to publishing the results, all tasks to be dealt with a lot of responsibility.

5.1 Guidelines of CCEE

The written examination should be of 60 minutes duration. It should consist of 40 objective questions. A typical objective type exam paper may contain the following types of questions:–

- Multiple choice
- Yes or No
- True or False

Objective type questions are useful in testing the recognition and recall abilities of students. They also help in keeping the exam short and easier to evaluate. In CCEE, there will be 40 objective type questions with 4 maximum answer options having only one correct option. The value of each objective type question is of one mark only. There will not be any negative marks for the wrong answers given by the students.

5.2 Guidelines for setting CCEE Question Papers

While setting the question papers for theory exam, the following weightage should be assigned depending on the difficulty level of the questions.

Levels	Requirements	Weightage
Level A - Easy	Requires elementary knowledge which may be obtained by attending all lectures and completion of mandatory lab assignments	25%
Level B – Medium	Requires thorough study of all course material, attendance at all lectures and completion of mandatory assignments	50%
Level C - Tough	Requires study and lab work beyond the prescribed course material and mandatory assignments	25%

5.3 Guidelines for generating CCEE questions

- Question paper setter has to use sample paper format provided by C-DAC
- Mention the module name without fail.
- Language of the question should be easy to understand.
- Answer options must have relevant objective type choices and only one correct answer.
- Questions must be prepared by referring appropriate books, reference books, and course material having good information.
- Questions must be created by the domain expert afresh and should not be copied directly from any book, website, existing previous question papers, etc.
- Question should be unique and should have not been published anywhere.

- Mention the source of the question wherever possible, as it may help in referring the same for detailing if required.
- The question paper should have questions covering the entire syllabus.
- The questions have to be typed in MS Word with “Arial” having letter size 12 point. Do not bold any letter, word or sentence in any part of the question paper.
- It is essential to give password to the word document and communicate the password separately.
- It is essential that utmost care is taken at your end to maintain the secrecy of the soft copy at all times.
- An expert team will review all questions. Questions will be filtered as per following:

    - If the question is incomplete
    - If the answer of the question is wrong
    - If the question is not part of the syllabus
    - If the question appears more than once
    - If the question is too lengthy
    - If the question is irrelevant
    - If the options to the questions are irrelevant.

5.4 Evaluation of CCEE

CCEE evaluations are machine based, wherein the response files are generated and sent to the evaluating centre in decrypted format.

If a student requests for re-evaluation then the student has to pay Rs. 200 /- for each module and it should be routed through training centre. The re-evaluation fee should be paid to the respective C-DAC training Centre, in case of Authorized Training Centres associated to C-DAC, Pune, payment to be made in favour of "C-DAC, ACTS" and payable at Pune. (This is applicable only for theory exam.)

5.5 Attendance

Overall 75% attendance is required for a student to be eligible for the CCEE.

6. Evaluation of Lab Exam should be based on the following criteria

Criteria	Details	Max Marks
Algorithm	Documentation of Algorithm and Flowchart	2
	Program adheres to the algorithm and flowchart	2
Efficiency	Program is using only the required number of variables /conditions/loops/pointers etc and is optimal	2
Correctness	The program produces desired output for a given input	20
	The program handles all valid and invalid inputs	
Software Engineering Principles	The program has meaningful variable/function names	2
	The program is commented properly (At least 20% of the code should be commented)	2
Viva	Questions based on the lab test	10
	Total Marks	**40**

7. Moderation

Grace Marks should be given to only deserving candidates as an exception and not as a rule. Grace marks would be awarded as per the methodology below:

a. Maximum of 4% of total centralised theory exam marks can be awarded to a candidate.
b. Maximum of 8% of individual module test marks (maximum marks) can be awarded per module.

Name of the course	Total Marks	Maximum grace marks for the course	Maximum Marks per module	Maximum grace marks per module
e-DAC	240	16	40	3.2

Grace marks should be applied only after final marks are compiled for each component of the evaluation (ie, Internal Assessment, Lab Exam, CCEE and Project). On completion of the moderation exercise the revised marks should be updated in the marks database.

If a student has cleared all the modules without availing any grace marks but is falling short by a few marks of attaining a better grade, then the competent authority may award additional grace marks at his/her discretion (subject to a maximum of 0.5% of total marks) on the aggregate total to enable the student to migrate to the next higher grade.

8. Re-examinations

The following conditions will be applicable for the CCEE re-exam:

- Students who do not appear for an exam on the scheduled date will not have an automatic right to re-examination. Only those students who, in the opinion of the Centre/Course Coordinator have a genuine reason for being absent may be allowed to appear for a re-exam.
- Students who have failed an exam may be allowed to appear for a re-exam.
- Re-exams should be conducted following the same process as the regular examination.
- Students, who failed/remained absent in the CCEE, shall be allowed to appear in the re-examination only once.
- Students who remain absent or fail in the re-examination will not get any further chance for appearing for the re-examination. In such case the candidate can receive the Performance Statement and the certificate of participation without any grade.
- On evaluation of their answer sheets 20% of the marks obtained by the students will be deducted (towards de-rating for re-examination) for arriving at the final score, i.e. in order to clear the module test the student has to score a minimum of 50% marks instead of 40%.
- The fee for the re-exam is currently NIL.
- There will be no re-exam after the re-exam

9. Evaluation Guidelines for Aptitude & Communication Module:

Total marks for evaluation will be 100 for Aptitude & Communication (Tests/ Presentation/ Seminar/ GD/ Attendance, etc). After evaluation, marks need to be converted to grades as per the scale given in this document, which will be mentioned in the mark sheet.

9.1 Evaluation Method

S. No.	Component Name	Marks	Total Marks
1	Aptitude Tests	50	

2	Overall Communication Skills	20	100
3	Presentation / Seminar/ GD	20	
4	Attendance for the module sessions	10	

The examination for this module must be conducted for at least that number of marks. However, the Centre may conduct evaluation for a higher number of marks, in which case the marks will be scaled down. For e.g. if the exam for presentation is conducted for 50 marks, the marks earned by the student will be scaled down to out of 20.

9.2 Guidelines for Presentation/Seminar/Group Discussion Evaluation

Evaluation of Presentation, Seminar and Group Discussion needs to be carried out as per the following guidelines. Presentation/Seminar evaluation for 50 marks will be segregated as follows:

Communication skills	10
Presentation skills	10
Flow of presentation	10
Contents of the presentation	10
Depth of knowledge (Q&A)	10

10. Software Project Module:

- Students in teams will be required to identify project topics in consultation with faculty members within the first three months of the course.
- Students may do industry-sponsored projects, but will be required to do the project work within given timeframe.
- The Software Project module is divided in two phases.

10.1 Phase I – SRS & Design

- *Tasks:* Project finalisation, requirements gathering, feasibility study, software design and project plan.
- *Deliverable*: Software Requirement Specification (SRS).
- *Schedule*: This phase will be executed along with the Software Development Methodologies sessions to enable better absorption of the concepts.

10.2 Phase II – Development & Testing

- *Tasks*: Coding and testing of the software system/application to be developed.
- *Deliverables*: Project report, functional software system/application.
- Schedule: This final phase will be executed during the last month of the course.

10.3 Project Evaluation

- In Phase I, students will present the design and plan on the schema of the project, and this will be evaluated by the project supervisor.
- In Phase II, a mid evaluation at the middle of the project development, and a final evaluation at the end of the project will be done in the presence of the project supervisor and an examiner.
- Weightage of the Project module will be as follows:
  - Phase I – 20%, Phase II – 80% (Mid Evaluation 20% + Final Evaluation 60%)
- Performance in the Project module will be awarded in grade based on the combined marks obtained all the evaluations of both the phases of the project.
- The Project grade will be mentioned separately on the Performance Statement and will have no effect on the overall grade obtained by a student.

11. Ensuring Security of Evaluation data/records

- Ensure that all data relating to evaluation of students is stored in a secure place that cannot be accessed by unauthorized personnel.
- All question papers must be prepared and stored in a separate area specifically designated for the purpose.
- Whenever any external faculty sets a question paper, ensure that he/she follows the guidelines given by C-DAC.
- Ensure that only one copy of any question paper is prepared in physical (printed) form for review and revision.
- When the question paper is finalized, print one master copy and get it signed by the paper setter, Reviewer and DRM Evaluation.
- The data relating to evaluation of students, such as soft copies of question papers and answer keys, student marks database and performance statements etc. must be kept in a separate domain/directory which is accessible only to authorized personnel. Ensure that the data is regularly backed up.
- The Centres according to guidelines provided by C-DAC will conduct the evaluation of the laboratory and internal assesments locally.

12. General guidelines for award of grades

The marks of all the 6 modules (each module with 100 marks) of e-DAC shall be added to get total marks out of 600. The course grades shall be awarded as mentioned in the below table.

%	Grade
85 and above	A+
70 to Less than 85	A
60 to Less than 70	B
50 to less than 60	C
40 to less than 50	D
Less than 40 (Fail)	F

Aptitude & Communication and Project modules will also be graded separately as per the above table.

Teaching Guidelines for

Concepts of Programming, Operating System & Software Engineering

Diploma in Advanced Computing (e-DAC)

September 2020

Duration: 36 theory hours + 28 lab hours (**64 hours**)

Evaluation: 100 Marks
Weightage: Theory exam – 40%, Lab exam – 40%, Internal exam – 20%

Part I - Basic Programming Concepts

Duration: 10 theory hours + 10 lab hours (**20 hours**)

Objective: To introduce the fundamental programming concepts in Java.

Prerequisites: Knowledge of computer fundamentals

Text Book:
- Core and Advanced Java Black Book / Dreamtech Press

References:
- Java The Complete Reference by Herbert Schildt / McGraw Hill
- Core Java : Fundamentals - Volume 1 Gary Cornell, Cay S. Horstmann/ Pearson
- Programming in Java by Sachin Malhotra, Saurabh Choudhary / Oxford University Press

---

(Note: Each Session is of 2 hours)

Session 1: Getting Started
Lecture:
- Setup development environment (JRE, JDK, eclipse)
- Writing your first Java program
- About main () method
- Constructor in Java

Lab:
Write Java programs to:
- Print Hello World
- Add two numbers/binary numbers/characters
- Calculate compound interest
- Calculate power of a number
- Swap two numbers

Session 2: Object Oriented Concepts
Lecture:
- Class & Object
- Access Specifier
- Java Data Types, Primitives and Binary Literals

Lab:
Write Java programs to:
- Calculate area of rectangle
- Calculate area and circumference of circle using multiple classes
- Java program to find ASCII value of a character

Session 3: Operators
Lecture:
- Arithmetic Operator
- Relational Operator
- Logical Operator
- Unary Operator
- Ternary Operator
- Assignment Operator

Session 4: Conditional and Looping Statements
Lecture:
- If, else if, switch
- break & continue keyword
- for loop
- while loop
- do while loop
- static & final keyword
- Recursion

Lab:
Write Java programs to:
- Display prime numbers between 1 and 100 or 1 and n
- Swap two variables without using the third variable
- Find the factorial of a number
- Check if a number is palindrome or not
- Print Fibonacci series till n
- Add two integer variables in 5 different ways using functions and control statement
- Find square root of a number without sqrt method
- Check Armstrong number
- Calculate grades of students using their marks
- Use switch case, recursion, print patterns, etc.

Session 5: Arrays
Lecture:
- Initializing an Array in Java
- Two dimensional array in java
- Java Variable Arguments explained
- Add, update, read array elements
- Sorting and searching in array
- Java String Array to String
- How to copy arrays in Java

Lab:
Write Java programs to:
- Calculate average of numbers using Array
- Reverse an array

- Sort an array in ascending order
- Convert char Array to String
- Add two Matrix using Multi-dimensional Arrays
- Sort strings in alphabetical order
- Find out the highest and second highest numbers in an array
- Concatenate two arrays

Part II - Operating System Concepts

Duration: 16 theory hours + 8 lab hours (**24 hours**)

Objective: To introduce Operating System concepts with Linux environment, and to learn Shell Programming

Prerequisites: Basic Knowledge of programming with object oriented concepts

Text Books:
- Operating Systems Principles by Abraham Silberschatz, Peter Galvin & Greg Gagne / Wiley

References:
- Modern Operating Systems by Andrew Tanenbaum & Herbert Bos/ Pearson
- Principles of Operating Systems by Naresh Chauhan / Oxford University Press
- Beginning Linux Programming by Neil Matthew & Richard Stones / Wrox
- Operating System : A Design-Oriented Approach by Charles Crowley / McGraw Hill

---

(Note: Each Session is of 2 hours)

Sessions 1 & 2:
Lecture:
Introduction to OS
- Evolution and components of Operating System
- Different from other application software
- Functionality and Services of Operating System
- Types of Operating System

Introduction to Linux
- Basics of File System types
- Commands associated with files/directories;
- Permissions (chmod,chown, etc)
- access control list

File Management
- Attributes and Operations on File Management
- File Access Methods
- Directory Structure

Lab:
Use various commands in Linux system: ls, cp, mv, lpr, sort, grep, cat, tac, more, head, tail, man, whatis, whereis, locate, find, diff, file, rm, mkdir, rmdir, cd, pwd, ln and ln –s, gzip, zip and unzip, tar and its variants, cal, bc, date, time, wc, touch, echo, who, finger, w, whoami, alias, unalias, touch, push, pop, jobs, ps, etc.

Session 3: Shell Programming
Lecture:

- Types of shells in Linux
- Shell Variables and Wild Card symbols
- Shell Meta characters
- Command line arguments
    - Read, Echo, decision loops, arithmetic expressions;

Lab:
Practice scripting on:

- Command line arguments
- Arithmetic in shell scripts
- Read and echo commands in shell scripts
- Taking decisions: if-then-fi , if-then-else-fi, case control structure

Session 4: Process
Lecture:

- Process States
- Preemptive and non-preemptive processes
- Process life cycle

Lab:
Create new system process using fork system call
Implement zombie and orphan processes

Sessions 5 & 6: Process scheduling algorithms with examples
Lecture:

- FCFS
- RR
- Shortest Job First
- Priority

Session 7: Threads
Lecture:

- Types of Threads - user and kernel threads
- Difference between Threads and Process

Session 8: Concurrency Control
Lecture:

- Deadlock Handling Strategies
- Deadlock Prevention
- Deadlock Avoidance

Part III - Software Engineering Concepts

Duration: 10 theory hours + 10 lab hours (**20 hours**)

Objective: To build knowledge of software development methodologies.

Text Book:
- Software Engineering by Chandramouli / Pearson

References:
- Software engineering by Ian Sommerville / Pearson
- Clean Code: A Handbook of Agile Software Craftsmanship by Robert C. Martin / Prentice Hall
- User Stories Applied: For Agile Software Development by Mike Cohn / Addison Wesley

---

(Note: Each Session is of 2 hours)

Session 1:
Lecture
- Developing an application in a team
- Issues developers face when working in a team
- Introduction to code versioning system
- Introduction to git
- Introduction to git repository and git structure
- Adding code to git
- Creating and merging different git branches

Lab
- Create a local git repository
- Commit the initial code
- Update the code
- Use git commands to
  - Get the updated files
  - List the changes
  - Create branch
  - Merge branch

Session 2:
Lecture
- Introduction to software engineering
- Software Development Life Cycles
- Requirements Engineering
- Design and Architectural Engineering
  - Design Models
  - UML
- Object Oriented Analysis and Design

Lab
- Prepare software requirement specification for web application
- Create the initial use-cases, activity diagram and ER diagram for the final project

Session 3:
Lecture

- Introduction to Agile development model
- Agile development components
- Benefits of Agile model
- Introduction to different tools used for agile web development
- Introduction to Atlassian Jira
    - Add Project
    - Add Tasks and sub-tasks
    - Create sprints with tasks
- Case study of developing web application using agile methodology

Lab

- Create different sprints in Atlassian Jira for different features

Session 4:
Lecture

- Introduction to software testing
- Principles of software testing
- Verification and validation
- Quality Assurance vs Quality Control vs Testing
- Introduction to STLC and V Model
- Types of testing: manual and automation
- Tools used for automation testing
- Introduction to testing methods: white-box, black-box and grey-box
- Introduction to functional and non-functional testing

Lab

- Create a test plan for project
- Document the use cases
- Create test case document for different sprints (designed in SE)

Session 5:
Lecture

- Introduction to Selenium (use Eclipse IDE)
- Load web driver
- Create selense commands: locators: by ID, name, class, tag name, XPath
- Add interactions: text box, radio button selection, check box selection, drop down item selection, keyboard actions, mouse actions, multi select

Lab

- Download and configure Selenium
- Create a test suite
- Add commands and interactions

Teaching Guidelines for

Object Oriented Programming with Java

Diploma in Advanced Computing (e-DAC)

September 2020

Duration: 44 theory hours + 44 lab hours **(88 hours)**

Objective: To reinforce knowledge of Object Oriented Programming concepts using Core Java.

Prerequisites: Basic knowledge of computer programming

Evaluation: Total 100 marks
Weightage: Theory exam – 40%, Lab exam – 40%, Internal exam – 20%

Text Book:
- Core and Advanced Java Black Book / Dreamtech Press

References:
- Java 8 Programming Black Book / Dreamtech Press
- Core Java : Volume 1 - Fundamentals by Cay S. Horstmann / Prentice Hall
- Core Java : Volume 2 - Advanced Features by Cay S. Horstmann / Prentice Hall
- Programming in Java by Sachin Malhotra, Saurabh Choudhary / Oxford University Press
- Java The Complete Reference by Herbert Schildt / McGraw Hill
- Core Java 8 for Beginners by Sharanam Shah, Vaishali Shah / Shroff Publishers
- Murach's Java Programming by Joel Murach / Mike Murach
- Object-Oriented Analysis and Design with applications by Grady Booch / Pearson
- Object-Oriented Analysis and Design Using UML - An Introduction to Unified Process and Design Patterns by Mahesh P. Matha / PHI

(Note: Each Session is of 2 hours)

Session 1:
Lecture:
Introduction to java
Features of java
JVM Architecture
JDK and its usage
Structure of java class
Working with data types: Primitive data types

Session 2:
Lecture:
Operators
- Unary, binary, Arithmetic, Assignment, compound, relational, logical, equality

Control statements
- if-else-if, switch, ternary operator, for loop, while loop, do-while loop

Declaring variables and methods
Data type compatibility

Lab:
Get yourself acquainted with java environment.
Print different patterns of asterisk (*) using loops (e.g. triangle of *).
Tutorial:
Compare syntactical similarities and dissimilarities between Java and C++.

Session 3:
Lecture:
Static variables and methods
Accessing static variables and methods of different class
Introduction to reference data types
Reference variables and methods
Difference between reference data types and primitive data types
Difference between reference variable and static variable

Session 4:
Lecture:
Constructors, initializing reference variables using constructors
Pass by value v/s pass by reference
Re-assigning a reference variable
Passing reference variable to method
Initializing reference variable of different class
Heap memory and stack memory

Lab:
Print default values of static & instance variables for different data types.
Build a class Employee which contains details about the employee and compile and run its instance.
Build a class which has references to other classes. Instantiate these reference variables and invoke instance methods.
Tutorial:
Understand role of stack and heap memory in method invocation and object creation.

Object Oriented Programming Concepts

Session 5:
Lecture:
Introduction to OOP concepts
Encapsulation
Inheritance: single & multilevel

Session 6:
Lecture:
Inheritance: Hierarchical
Polymorphism: Compile time and runtime polymorphism
Rules of overriding and overloading of methods
super and this keywords
Upcasting & downcasting of a reference variable

Lab:
Create a class Employee and encapsulate the data members.
Create demo applications to illustrate different types of inheritance.

Session 7:
Lecture:
Abstract class and abstract methods
Interface (implementing multiple interfaces)

Session 8:
Lecture:
Final variables, final methods and final class
Functional interface
New interface features (Java 8 & above)
Arrays
Enumerations

Lab:
Create an Array of Employee class and initialize array elements with different employee objects.
Try to understand the no of objects on heap memory when any array is created.

Session 9:
Lecture:
Access modifiers (public, private, protected and default)
Packages and import statements
Static imports
Constructor chaining (with and without packages)
Accessing protected variables and methods outside the package

Session 10:
Lecture:
Garbage collection in java
Requesting JVM to run garbage collection
Different ways to make object eligible for garbage collection: (Nulling a reference variable, Re-assigning a reference variable & island of isolation)
Finalize method

Lab:
Create a demo application to understand the role of access modifiers.
Implement multilevel inheritance using different packages.
Access/invoke protected members/methods of a class outside the package.
Override finalize method to understand the behavior of JVM garbage collector.

Sessions 11 & 12:
Wrapper Classes and String Class
Lecture:
Wrapper classes and constant pools
String class, StringBuffer & StringBuilder class
String pool

Lab:
Create sample classes to understand boxing & unboxing.
Use different methods of java defined wrapper classes.
Create StringDemo class and perform different string manipulation methods.
Tutorial:
Understand the difference between String / StringBuffer / StringBuilder.

Sessions 13 & 14:
Exception Handling
Lecture:
Exception hierarchy, Errors, Checked and un-checked exceptions
Exception propagation
try-catch-finally block , throws clause and throw keyword
Multi catch block
Creating user defined checked and unchecked exceptions
Lab:
Create user defined checked and unchecked exceptions .

Session 15:
java.io & java.nio Package
Lecture:
Brief introduction to InputStream, OutputStream, Reader and Writer interfaces
NIO package
Serialization and de-serialization
Shallow copy and deep copy
Lab:
Create a Demo class to Read & write image/text files.
Create SerializationDemo class to illustrate serialization and de-serialization process.

Session 16:
Lecture:
Object Class & java.util Package
Date, DateTime, Calendar class
Converting Date to String and String to Date using SimpleDateFormat class
Object Class: Overriding to String, equals & hashcode method

<u>Collections</u>

Session 17:
Lecture:
Introduction to collections: Collection hierarchy
List, Queue, Set and Map Collections
List Collection:

- ArrayList, LinkedList
- Vector (insert, delete, search, sort, iterate, replace operations)

Collections class
Comparable and Comparator interfaces
Queue collection

Lab:
Create DateManipulator class to convert String to date, date to String and to find out number of days between two dates.
Create a List of java defined wrapper classes and perform insert/delete/search/iterate/sort operations.
Create a collection of Employee class and sort objects using comparable and comparator interfaces.
Implement Queue data structure using LinkedList and Queue collection.

Sessions 18 & 19:
Lecture:
Set Collection:

- HashSet, LinkedHashSet & TreeSet collection
- Backed set collections

Map Collection:

- HashTable, HashMap, LinkedHashMap & TreeMap classes
- Backed Map collections

Generics
Concurrent collections

Lab:
Create an Employee HashSet collection and override equals & hashCode methods to understand how the set maintains uniqueness using these methods.
Create a Sample class to understand generic assignments using "? extends SomeClass", "? super someclass" and "?".

Session 20:
Lecture:
MultiThreading : Thread class and Runnable Interface
sleep, join, yield, setPriority, getPriority methods
ThreadGroup class

Lab:
Invoke private methods of some other class using reflection.
Create multiple threads using Thread class and Runnable interfaces.
Assign same task and different task to multiple threads.
Understand sleep, join, yield methods.

Sessions 21 & 22:
Lecture:
Synchronization
Deadlock
Wait, notify and notifyAll methods
Inner classes

Lab:
Create a Deadlock class to demonstrate deadlock in multithreading environment.
Implement wait, notify and notifyAll methods.
Demonstrate how to share threadlocal data between multiple threads.
Create multiple threads using anonymous inner classes.

Teaching Guidelines for

Algorithms and Data Structures

(Using Java)

Diploma in Advanced Computing (e-DAC)

September 2020

Duration: 32 theory hours + 28 lab hours (**60 hours**)

Objective: To reinforce knowledge of problem solving techniques, data structure concepts and analysis of different algorithms using Java.

Prerequisites: Knowledge of programming with object oriented concepts

Evaluation: 100 Marks
Weightage: Theory exam – 40%, Lab exam – 40%, Internal exam – 20%

Text Book:
- Fundamentals of Data Structures in C++ by Horowitz, Sahani & Mehta / Orient Longman

References:
- Problem Solving: Best Strategies to Decision Making, Critical Thinking and Positive Thinking by Thomas Richards / Kindle Edition
- Data Abstraction and Problem Solving with Java: Walls and Mirrors by Janet Prichard , Frank M. Carrano / Pearson
- Data Structures, Algorithms and Applications in C++ by Sartaj Sahni
- Object-oriented Analysis and Design Using UML - An Introduction to Unified Process and Design Patterns by Mahesh P. Matha / PHI
- Introduction to Algorithms by Cormen, Leiserson, Rivest and Stein

---

(Note: Each Session is of 2 hours)

Sessions 1 & 2:
Problem Solving & Computational Thinking
Lecture:
- Define the problem
- Identify the problem
- Introduction to Problem Solving
- Problem solving basics

Lab:
- Faculties need to assign different problems, mostly real world problems
- Students (by team wise, there are two students in a team) need to analyze as per the techniques learned
- Based on the above problems students need to select as per the selection criteria learned
- They need to implement the selected solution and need to do the documentations.

Sessions 3 & 4:
Algorithm & Data Structures
Lecture:
Objective of the session: At the end of the session students should know, what is the importance of data structure in problem solving. How stacks, queues, circular queues work. Their real world applications. How to solve problems using these data structures.

- Introductory Concepts
- Algorithm Constructs
- OO design: Abstract Data Types (ADTs)
- Basic Data Structures
  - Arrays
  - Stacks
  - Queues
  - Circular Queues

Lab:

- Implement Stack through Array
- Implement Queues with inserting element at different location (First, Last and at specific location)
- Implement circular queue

Sessions 5 & 6:
Linked List Data Structures
Lecture:
Objective of the session: At the end of the session students should know, what are applications of Linked List, different types of link list. Comparison with arrays as when to use linkedlist and when to use array.

- Linkedlists
  - Singly LinkedLists
  - Doubly LinkedLists
  - Circular LinkedLists
  - Node-based storage with arrays

Lab:

- Implement circular queue using linked list
- Implement stack using using linked list

Session 7:
Recursion
Lecture & Lab:
Objective of the session: At the end of the session students should know what is recursion, type of recursion, local variable in recursion, stack manipulation during recursion, function complexity

- What is recursion?
- What is the base condition in recursion.
- Direct and indirect recursion.
- Memory is allocated to different function calls in recursion.
- Pro and cons of recursion
- Function complexity during recursion

Sessions 8 & 9:
Trees & Applications
Lecture:
Objective of the session: At the end of the session students should know what is the use of binary trees, how to create binary search trees. Different tree traversals. What are the applications of binary trees? How to calculate search complexity in binary search trees? What are the limitations of binary search trees? What are different options to overcome the binary search tree limitations.

- Introduction to trees
- Trees and Terminology
- Tree Traversals
- Binary trees
- Complete binary trees / Almost complete binary tree (ACBT)
- Array Implementation of ACBT
- Binary search trees
- AVL Tree
- Multi-way Tree

Lab:

- Write a program to implement a binary search tree and the following operations onit:
  - Create()
  - Tree traversals (Breadth First Search, Depth First Search, Inorder(), Preorder(), Postorder())
  - Delete()

Sessions 10 & 11:
Searching & Sorting Algorithms
Lecture:
Objective of the session: At the end of the session students should know what are the different types of sorting and searching algorithms, why all the sorting algorithms are equally important despite different time/space complexity of the algorithms. How the complexity is calculated for each of them. How to pick a sorting algorithm given the nature of the data to be sorted.

- Objectives of Searching
  - The Sequential Search
  - Analysis of Sequential Search
  - The Binary Search
- Analysis of Binary Search
- Introduction to sorting
  - Selection sort
  - Insertion sort
  - Bubble sort
  - Shell Sort
  - Heapsort
  - Mergesort
  - Quicksort
- Analysis of sorting algorithms

Lab:

- Writing program to search an item through sequential search technique.
- Implement to find an item in a list through binary search

- Implement sorting algorithm for: insertion sort, quicksort

Session 12:
Hash Functions and Hash Tables
Lecture:
Objective of the session: At the end of the session students should know what is hashing, what is the importance of hashing, comparative complexity of hashing with other search techniques. Problems (collision) with hashing and what are the different solutions of that.

- Hashing & Introduction to hashtables
- Hash Functions
- Different type of hash functions
- Collision Resolution
- Clustering
  - Primary
  - Secondary
- Linear Probing
- Quadratic Probing
- Double Hashing
- Inserting and Deleting an element from a hash table

Lab:

- Implement hashing techniques in different programs solved earlier
- Write a program to implement Hashtable

Sessions 13 & 14:
Graph & Applications
Lecture:
Objective of the session: At the end of the session students should know what is graph? Why is graph the most generic data structure? Different types of graphs. Different representation of graph? Graph traversals (Breadth First Traversal, Depth First Traversal). Different applications which can be solved with graphs, real world and programming problems with graphs.

- Introduction to graph theory
- Graph Terminology
- Different types of Graphs
- Representation of Graphs
  - Adjacency Matrix
  - Adjacency List
  - Graph Traversal Algorithms (Breadth First Search, Depth First Search)
- Shortest Path
  - Level Setting : Dijkstra's algorithm
  - Level Correcting: All-pairs shortest path, Floyd-Warshall algorithm
- Spanning Trees
  - Minimum spanning tree algorithms,
  - Prim's algorithm
  - Kruskal's Algorithm

Lab:

- Implement a graph using adjacency Matrix and traverse using Depth FirstSearch.
- Implement a graph and do traversal using stack and queue.

Sessions 15 & 16:
Algorithm Designs
Lecture:

Objective of the session: At the end of the session students should know what are different classes of algorithms. What is the nature of each class of algorithms? How to pick an algorithm for a particular problem. What problems fall under each class of algorithms. What are the worst case, average case and the best case for algorithms?

- What are the different class of algorithms
- How to write efficient Algorithm
- Introduction to algorithm design techniques
- Algorithm Design techniques
- Analysis of an Algorithm
  - Asymptotic Analysis
  - Algorithm Analysis
- Analysis of different type of Algorithms
  - Divide and Conquer Algorithm
  - Greedy algorithm
  - Dynamic Programming algorithm
  - Brute force algorithm
  - Backtracking algorithms
  - Branch-and-bound algorithms
  - Stochastic algorithms
- Complexity
  - Complexity Analysis
  - Space complexity of algorithm
  - Time complexity of algorithm
- Case study on Algorithm Design techniques
- Application of Data structures

Assignment – Read:

- Study on different Algorithms
- Compare different Algorithms previously programmed and do the analysis

`

Teaching Guidelines for

Database Technologies

Diploma in Advanced Computing (e-DAC)

September 2020

Duration: 30 theory hours + 30 lab hours (**60 hours**)

Objective: To introduce students to RDBMS and NoSQL Databases and facilitate hands-on experience on SQL (Using MySQL) and MongoDB.

Prerequisites: Working Knowledge of Windows and Linux, Familiarity with Programming and Object Oriented concepts.

Evaluation: 100 Marks
Weightage: Theory Exam – 40%, Lab exam – 40%, Internal exam – 20%

Text Book:
- Murach's MySQL by Joel Murach / Shroff Publisher

References:
- Database System Concepts by Abraham Silberschatz, Henry Korth and S. Sudarshan / McGraw Hill
- Database Design and Relational Theory: Normal Forms and All That Jazz by C. J. Date (Author) / O'Reilly
- Fundamentals of Database System by Shamkant B. Navathe, Ramez Elmasri / Pearson
- MySQL: The Complete Reference by Vikram Vaswani / McGraw Hill
- SQL & NoSQL Databases: Models, Languages, Consistency Options and Architectures for Big Data Management by Andreas Meier and Michael Kaufmann / Springer
- MongoDB: The Definitive Guide by Shannon Bradshaw, Eoin Brazil and Kristina Chodorow / O'Reilly
- http://bigdata.stratebi.com/?language=en

(Note: Each Session is of 2 hours)

Session 1:
Lecture
Introduction to DBMS, Basic Database Terminology
Database Storage Architecture, Concept of Database Instance and Schema, Distributed Databases
Types of DBMS: Relational, Object Relational and NoSQL Databases
Data Models (Conceptual, Logical, Physical), Codd’s 12 rules for RDBMS
Database Design, Entity-Relationship Diagram (ERD)

Lab
Using MySQL Monitor, Shell and Workbench
ER Diagrams to Relational Table mapping

Session 2:
Lecture
Introduction to SQL, Categories of SQL Commands: DDL, DML, DCL, DTL/TCL.
Introduction to MySQL, MySQL Clients (Monitor, Shell, Workbench).
SQL/MySQL Data Types, Database Constraints (Primary Key, Unique, Not Null, Foreign Key, Default, Check)

Lab (3 hrs)
Performing basic CREATE, SELECT, INSERT, UPDATE, DELETE, DROP operations on Tables

Session 3:
Lecture
Normalization, Need for Normalization
Data Redundancy, Data Anomalies, Functional Dependency
Normal Forms (1NF, 2NF, 3NF, BCNF) with examples, Introduction to 4th and 5th NF, Need of Denormalization

Lab
Perform 1NF, 2NF, 3NF, BCNF

Session 4:
Lecture
LIKE Operator, DISTINCT, Sorting (Order by clause).
BETWEEN, AND, OR Operators, Comparing Nulls (IS NULL/IS Not NULL), IN/NOT IN
Relational Algebra Operations (Selection, Projection, Union, Intersect, Minus)

Lab
Using Like, Distinct, Order By, Between...And
Comparing Nulls, Using IN/Not-In
Union/Union ALL

Session 5:
Lecture
Aggregate Functions
Grouping Things Together (Group By, Having)

Lab
Defining Data Types for Columns
Creating, Altering, Dropping Constraints
Aggregate Functions: SUM(), AVG(), COUNT(), MAX(), MIN(), COUNT()
Using Group By, Having Clause

Session 6:
Lecture
Joins (Equi, Inner, Outer, Natural, Cross), SQL Standard Syntax for Joins
Copying table structure/data, Sequences (AUTO_INCREMENT)

Lab
Queries on Various type of Joins using OLD and SQL Standard Syntax
Copying table structure, Copying data from one table to another
Using AUTO_INCREMENT

Session 7:
Lecture
Subquery, Correlated Subquery, EXISTS/NOT EXISTS
TCL Commands (Commit/Rollback/SavePoint), DCL Commands (GRANT/REVOKE/GRANT OPTION)

Lab (3 hrs)
SubQueries, Correlated Queries
Using Exists/Not-Exists
Using Commit/Rollback/Savepoint
Granting/revoking privileges on database objects

Session 8:
Lecture
Views, Types of Views, Simple and Complex Views
Indexes, Benefit of Indexes, Type of Indexes, Temporary Tables
MySQL Storage Engines (InnoDB, MyISAM and others),
ACID Properties, Concurrency and Locks

Lab
Creating Views, Querying using Views
Creating Indexes
Creating Temporary Tables
Database Locks

Session 9:
Lecture
Introduction to MySQL Programming, Use of MySQL Programs,
Introduction to Stored Procedures, Benefits of Stored Procedures
Procedure Parameters (IN, OUT and INOUT)

Lab
Creating procedure without parameters
Creating Procedure with (IN/OUT/INOUT) Parameters

Session 10:
Lecture
Flow Control Statements (LOOP, WHILE and REPEAT)
Using above statements in Stored Procedures/ Functions
Conditional Statements (IF, IF-ELSE-THEN, SWITCH CASE)
Example of each type of statement

Lab
Use of flow control statement in Stored Procedure
Use of conditional statements in Stored Procedure

Session 11:
Lecture
Loop constructs (ITERATE, LEAVE)
Functions with and without parameters
MySQL Built-in functions (string, numeric, date etc.)

Lab
Creating Function and returning value from it
Use of built-in functions in queries

Session 12:
Lecture
Cursors (Asensitive, Insensitive, Read only, Nonscrollable)
Cursors example and real time use
Triggers (BEFORE, AFTER), New and Old trigger variables
Trigger Examples and real time use
Full-Text Search, Pattern Matching with RegEx

Lab
Writing procedures with Declare, fetch and close cursor
Example of each type of cursors
Create Triggers
Creating FULLTEXT Indexes for Full-Text Search
Using RegEx in queries

Sessions 13 & 14:
Lecture
Introduction to NoSQL database, Features of NoSQL Database
Structured vs. Semistructured and Unstructured Data
Difference between RDBMS and NoSQL databases,
CAP Theorem, BASE Model
Categories of NoSQL Databases
Introduction to MongoDB, Features of MongoDB
MongoDB command interface and MongoDB compass
MongoDB Documents & Collections
RDBMS & MongoDB analogies: relations/tables => collections; tuples/records => documents
JSON and BSON documents
Performing CRUD (CREATE, READ, UPDATE, DELETE) Operations, UPSERT

Lab (2 hrs)
Using MongoDB Shell and Compass
Creating Collections in MongDB
Performing Basic CRUD operations

Session 15:
Lecture
MongoDB Operators, Sorting and Indexing in MongoDB
Migrating from RDBMS to NoSQL

Lab
Complex Searching Using MongoDB Operators
Sorting data
Creating and and using indexes
Migrating from MySQL to MongoDB and Vice versa

Teaching Guidelines for

Web Programming Technologies

Diploma in Advanced Computing (e-DAC)

September 2020

Duration: 44 theory hours + 44 lab hours (**88 hours**)

Objective: To introduce the students to HTML, CSS, XML, JavaScript, jQuery, JSON, Ajax, Node.js, Express.js, React, React-Redux,and practical relevance of all these technologies.

Evaluation: 100 marks
Weightage: Theory Exam – 40%, Lab exam – 40%, Internal exam – 20%

Text Books:

- Fundamentals of Web Development, 1e, by Randy Connolly, Ricardo Hoar / Pearson
- MERN Quick Start Guide – Build web applications with MongoDB, Express.js, React, and Node by Eddy Wilson IriarteKoroliova / Packt

References:

- Internet & World Wide Web : How to Program by Paul Deitel, Henry Deitel&Abbey Deitel / Pearson Education
- XML - How to Program by Deitelet al /Pearson Education
- Ajax in Action by Dave Crane, Eric Pascarello /Dreamtech Press
- JavaScript: The Good Parts by Douglas Crockford / O'Reilly
- Pro MERN Stack: Full Stack Web App Development with Mongo, Express, React, and Nodeby Vasan Subramanian / Apress
- Web Application Security: A Beginner's Guide by Bryan Sullivan & Vincent Liu / Tata McGraw Hill
- W3Schools Tutorials [https://www.w3schools.com/]
- Mozilla Developer Network Web Development Tutorials [https://developer.mozilla.org/en-US/docs/Learn/Getting_started_with_the_web]
- Curated Tutorial Links on ES6, React, etc. [https://github.com/markerikson/react-redux-links]

(Note: Each Session is of 2 hours)

Session 1: Architecture of Web
Lecture:

- Brief history of the Internet
- How does the Internet work?
- Internet Protocol; HTTP
- Domain Names; Domain Name Service servers
- HTTP Protocols
  - Difference between HTTP1.0, HTTP 1.1, and HTTP 2.0
  - Methods – GET, POST, HEAD, PUT, DELETE, etc.
  - Status codes
  - Stateless nature of the protocol and HTTP Session
  - HTTPS
- Architecture of the Web

- Web servers – IIS, Apache server

Lab:

- Exploring different browsers
    - Mozilla Firefox, Google Chrome, Safari
- Exploring different text editors
    - Windows: Notepad++, Linux: Gedit or Vim or Emacs

Sessions 2-3: HTML (3 hrs)

Lecture:

- Introduction to HTML
- Document Object Model (DOM)
- Basic HTML Tags
    - Alignment, Headings, Anchor, Paragraph, Image, Lists, Tables, and iFrames
- HTML5
    - New features in HTML5
    - New elements, new attributes, link relations, microdata, ARIA accessibility
    - HTML5 Validation
    - Audio & Video Support
- HTML Forms & Controls
    - Input, Text Area, Radio Button, Checkbox, Dropdown, Submit, Reset, Button, etc.

Lab:

- Create a HTML form for building a resume.

Sessions 3-4: Cascading Style Sheets (CSS) (3 hrs)

Lecture:

- Introduction to CSS, Styling HTML with CSS, Structuring pages with CSS,
- Inline CSS, Internal CSS, External CSS, Multiple styles, CSS Fonts
- CSS Box Model
- id Attribute, class Attribute
- HTML Style Tags
- Linking a style to an HTML document

Lab:

- Apply inline, internal and external CSS to change colors of certain text portions, bold, underline, and italics certain words in the previously created HTML resume form.

Session 5: Responsive Web Design

Lecture:

- Introduction of UI Scripting
- The Best Experience for All Users
    - Desktop, Tablet, Mobile
- Bootstrap
    - Overview of Bootstrap, Need to use Bootstrap
    - Bootstrap Grid System, Grid Classes, Basic Structure of a Bootstrap Grid
    - Typography
    - Components – Tables, Images, Jumbotron, Wells, Alerts, Buttons, Button Groups, Badges/Labels, Progress Bars, Pagination, List Groups, Panels, Dropdowns, Collapse, Tabs/Pills, Navbar
    - Forms, Inputs
    - Bootstrap Themes, Templates

Lab:

- Update the design of the Resume form using Bootstrap

Session 6: JavaScript
Lecture:

- Introduction to JavaScript
- Variables in JavaScript
- Statements, Operators, Comments, Expressions, and Control Structures
- JavaScript Scopes
- Strings, String Methods
- Numbers, Number Methods
- Boolean Values
- Dates, Date Formats, Date Methods
- Arrays, Array Methods

Lab:

- Practice writing basic JavaScript programs for better understanding of the language constructs

Session 7: JavaScript
Lecture:

- Objects, Object Definitions, Object Properties, Object Methods, Object Prototypes
- Functions, Function Definitions, Function Parameters, Function Invocation, Function Closures
- Object Oriented Programming
    - Method, Constructor, Inheritance, Encapsulation, Abstraction, Polymorphism

Lab:

- Write a JavaScript program to sort a list of elements by implementing a sorting algorithm.
- Write a JavaScript program to list the properties of a JavaScript object.

Sessions 8 & 9: JavaScript
Lecture:

- Document Object Model (DOM)
    - Object hierarchy in JavaScript
    - HTML DOM, DOM Elements, DOM Events
    - DOM Methods, DOM Manipulation
- Forms, Forms API, Forms Validation
- Regular Expressions
- Errors, Debugging
- Introduction to Browser Dev Tool
- Pushing code quality via JSLint tool

Lab:

- Write a JavaScript function to get First and Last name from the previously created Resume form
- Validate the entire Resume form using client-side JavaScript
- Write a JavaScript function to validate whether a given value is RegEx or not.

Session 10: jQuery
Lecture:

- Introducing to jQuery
- jQuery selectors
- jQuery events
- jQuery animation effects
- jQuery DOM traversal and manipulation
- Data attributes and templates

- jQuery DOM utility functions
- jQuery plugins

Lab:

- Write a jQuery program to get a single element from a selection of elements of a HTML page.
- You are having sample data for the link. Write jQuery code to change the hyperlink and the text of an existing link.
- Write a jQuery program to attach a click and double-click events to all <p> elements.
- Write a jQuery program to hide all headings on a page when they are clicked.
    - Also find the position of the mouse pointer relative to the left and top edges of the document.

Session 11: JavaScript Object Notation (JSON)

Lecture:

- Introduction and need of JSON
- JSON Syntax Rules
- JSON Data - a Name and a Value,
- JSON Objects, JSON Arrays, JSON Files
- JSON parsing

Session 12: Ajax

Lecture:

- Introduction to Ajax
- Ajax Framework
- Ajax Architecture
- Web services and Ajax
- Ajax using JSON and jQuery

Labs:

- Create a page showing live score/feed using Ajax and JSON from a live sport/news service end-point given by the faculty

Session 13: Introduction to Node.js

Lecture:

- Introduction to Node.js
- Browser JS vs. Node.js
- ECMAScript 2015 (ES6)
- Node.js REPL

Lab:

- Install Node.js 12.x.x LTS version on your machine
- Write a recursive function in Node.js
- Write a Node program that prints all the numbers between 1 and 100, each on a separate line. A few caveats:
    - if the number is divisible by 3, print "foo"
    - if the number is divisible by 5, print "bar"
    - if the number is divisible by both 3 and 5, print "foobar"

Sessions 14 & 15: Node.js Asynchronous Programming

Lecture:

- Introduction to Asynchronous programming and callbacks
- Promises and async & await

- The Event Loop and Timers

Lab:
- Assignment on JavaScript callback functions
- Assignment on Timers, Promises, and Async & Await

Session 16: Node.js Modules
Lecture:
- Understanding Node modules, exports, and require
- Introduction to npm
  - package.json and package-lock.json files
  - Install, update, and manage package dependencies
  - Local and global packages

Lab:
- Create a module and import it in other programs
- Install a module/package using npm

Session 17: Node.js Modules – *fs* and *http*
Lecture:
- File I/O – Sync & Async Methods
- HTTP Module – Building an HTTP server
- Developing a Node web application

Lab:
- Write a program to create a new file and write some content to it in synchronous mode and read and display file contents on standard output in async mode
- Build a simple Node.js web application serving both HTTP GET and POST methods

Session 18: React
Lecture:
- Introduction to React
- React Elements and React Components
- Function and Class Components
- Working with React Components and Props
  - Compose components
  - Render components
  - Declutter components

Lab:
- Rebuild any previous plain HTML lab assignment using React
- Build a React Clock app showing time (hh:mm:ss) of any three countries

Session 19: React
Lecture:
- Introduction to State and Lifecycle
- Stateful components and lifecycle methods
- Props vs. State vs. Context
- Handling events
- Conditional rendering

Lab:
- Implement the following items in the React Clock app
  - Update the time (hh:mm:ss) using State and Lifecycle methods
  - Add a close function on each rendered clock component

    - Assign background color of rendered clock components based on AM, PM

Session 20: React
Lecture:
- Lists and Keys
    - Rendering Multiple Components
    - Basic List Component
- Working with forms and inputs
- Refs and the DOM
- Lifting state up

Lab:
- Implement and integrate a new feature in the React Clock app where one can select a country time zone from dropdown list and click on "Add" button to render it.

Session 21: React
Lecture:
- Error Boundaries
- Composition vs. Inheritance
    - Containment
    - Specialization
- Thinking in React

Lab:
- Implement error boundaries at appropriate places in the React Clock app

Session 22: Introduction to React-Redux
Lecture:
- Introduction to Redux
- Actions, Reducers, and Stores
- Usage with React

Lab:
- Make necessary changes in the design and implementation of React Clock app using React-Redux to maintain the application state.

Teaching Guidelines for

Web-based Java Programming

Diploma in Advanced Computing (e-DAC)

September 2020

Duration: 42 theory hours + 38 lab hours **(80 hours)**

Objective: To learn advanced concepts in java programming, and perform web Programming using Java.

Prerequisites: Knowledge of core Java programming.

Evaluation: Total 100 marks
Weightage: Theory exam – 40%, Lab exam – 40%, Internal exam – 20%

Text Book:
- Core and Advanced Java Black Book / Dreamtech Press

References:
- Servlet and JSP: A Tutorial by Budi Kurniawan / Brainy Software
- Spring in Action by Craig Walls / Manning Publications
- Advanced Java programming by Uttam K Roy / Oxford University press
- Sun Certified Enterprise Architect for Java EE Study Guide by Mark Cade & Humphrey Sheil / Pearson Education
- Professional Java EE Design Patterns by Murat Yener, Alex Theedom & Reza Rahman / Wrox

(Note: Each Session is of 2 hours)

Sessions 1 & 2
Lecture:
J2EE Overview
- J2EE Container
- Packaging Web applications
- J2EE compliant web application
- Deployment tools.
- Web application life cycle
- Deploying web applications.
- Web Services Support

JDBC & Transaction Management
- Introduction to JDBC API
- JDBC Architecture
- JDBC Drivers
- JDBC Classes & Interfaces: Driver, Connection, Statement, PreparedStatement, ResultSet and their relationship to provider implementations
- Stored procedures and functions Invocation
- SQL Injection overview and prevention

- Design Pattern: Data Access Object Pattern

Lab (2 hrs):

- Perform database CRUD operations using JDBC classes and interfaces.

Sessions 3, 4 & 5

Lecture:

- Servlets: Dynamic Content Generation
- Advantages of Servlets over CGI
- Servlet Life cycle
- Servlet API & Deployment
- Servlet Annotations
- The Servlet interface
- The HttpServlet, HttpServletRequest, HttpServletResponse
- Exception Handling
- Servlet, DAO, POJO DB Layers
- Session
- Session Management
- Session Tracking with
    - Cookies
    - HttpSession
- Request Dispatcher
- Page Navigation
- Complete Case study Servlet Based

Lab:

- Installing a servlet container (Tomcat)
- Adding Server to IDE
- Develop a structured dynamic web application (e.g. Library Management System) using servlets, deploy it in Tomcat
- Use HTTP Session in the Air Ticket Reservation System

Reading: Know more about the HTTP protocol at www.w3c.org
Tutorial: Compare which way of session tracking is better Cookies or HttpSession.

Sessions 6 & 7:

Lecture

- JSP: Separating UI from Content generation code
- MVC architecture
- Design Pattern: MVC Pattern
- Life cycle of a JSP page
- Directives, Implicit and Explicit Objects, Scriptlets, Expressions, Expression Language
- Scope
- JSP Error Page handling
- JSTL

Lab:

- Separate UI code from the controller code in your Library Management System by incorporating JSP and Servlets.
- Complete the implementation of Air Ticket Reservation System.

- Implement MVC based web application using Servlet, JSP

Sessions 8, 9 & 10:
Lecture:

- Hibernate Framework
    - Introduction to Hibernate Framework
    - Architecture
- Hibernate in IDE
    - Creating web application using Hibernate API
    - Lifecycle of Hibernate Entities
- HB with annotation example
- Hibernate Mappings and Relationships
- Collection and Component Mapping
- HQL, Named Queries, Criteria Queries

Lab:

- Demonstrate Hibernate as standalone library in Java application
- Develop a web application (Online Bookshop) using Hibernate Persistence

Reading: Study Hibernate architecture from **www.hibernate.org/docs**

Sessions 11, 12 & 13:
Lecture:

- What is Spring Framework
- Overview of Spring Architecture
- Spring MVC architecture
- Spring Modules Overview
- Understanding Spring 4 annotations (Basic Introduction)
- What is IoC (Inversion of Control)
- IOC container
- Dependency Injection
- Spring Beans
- Autowiring Beans
- Bean Scopes
- Spring MVC
- Model, Model & View, HandlerMapping, ViewResolver
- Design Pattern: Front Controller Pattern
- Spring MVC Web application with JSP views (without Spring Boot)
- Using Thymleaf as alternate View Technology (only introduction)
- Spring Validations
- Spring i18n, Localization, Properties
- File Upload example

Lab:

- Design and deploy Library Management System using Spring Web

Session 14 & 15:
Lecture:

- Spring Boot essentials

- Why Spring boot
- Spring Boot Overview
- Basic Introduction of MAVEN
- Building Spring Web application with Boot
- Spring Boot in detail (Use Spring Boot for all demo & assignments here onwards)
- Running a web application using Spring Boot with CRUD (with Static Data not DB)
- Spring Data JDBC

Lab:

- Create Hello World Spring Boot Web application
- Check Libraries importedby Spring Boot
- Create Spring Boot CRUD application with Thymeleaf as View technology and Spring JDBC

Sessions 16 & 17:
Lecture:
Spring Data Module

- Spring Data JPA (Repository support for JPA)
- CrudRepository & JPARepository
- Query methods
- Using custom query (@Query)

Lab:

- Add CRUD operations with Spring JPA etc. to earlier Spring Web application.

Sessions 18:
Lecture:
Spring AOP

- AOP Overview
- Spring AOP
- AOP Terminology and annotations: Advice, Join Points, Pointcuts, Aspects

Lab:

- Modify earlier Spring MVC application to Log all the requests using AOP

Sessions 19 & 20:
Lecture:
Building REST services with Spring

- Introduction to web services
- SOAP Vs RESTful web services
- RESTful web service introduction
- Create RESTful web service in java using Spring Boot
- RESTful web service JSON example
- RESTful web service CRUD example
- Using POSTMAN client to invoke REST API's
- REST service invocation using REST Template

Lab:

- Create REST API for Employee Management using Spring Boot
- Invoke it from POSTMAN app
- Invoke it from another Spring Boot Web application using REST Template

Session 21:
Lecture + Lab: (2 hrs)

- Testing in Spring
- Unit Testing of Spring MVC Controllers:
- Unit Testing of Spring Service Layer
- Integration Testing of Spring MVC Applications: REST API
- Unit Testing Spring MVC Controllers with REST

Teaching Guidelines for

Aptitude & Communication

Diploma in Advanced Computing (e-DAC)

September 2020

Duration: 60 theory hours + Practice sessions

Objectives: To reinforce knowledge of general aptitude
To speak in English confidently
To learn good writing and presentation skills
To prepare for and succeed in Interviews

Prerequisites: Knowledge of Mathematics & English.

Evaluation: Grading based on Tests, Writings, Presentations, Activities & Sessions
Weightage: Aptitude Tests – 50%, Communication – 50%

Aptitude Reference Books:
- Quantitative Aptitude by RS Aggarwal / S Chand
- Verbal & Non-Verbal Reasoning: RS Aggarwal / S Chand
- Quantitative Aptitude - Quantum CAT : Sarvesh K Verma / Arihant
- How to prepare GRE by Barron's / Galgotia
- Magic Book on Quicker Math by Manoj Tyra / BSC

Website to refer: www.indiabix.com

Communication Reference Books:
- Professional Communication Skills by AK Jain, PSR Bhatia & AM Shaikh / S. Chand
- Communication Skills by Sanjay Kumar & Pushp Lata / Oxford
- High School English Grammar & Composition by Wren & Martin / S. Chand
- English is Easy by Chetan Anand Singh / BSC
- Oxford Guide to English Grammar by John Eastwood / Oxford
- Business Communication by H S Mukerjee / Oxford
- Effective Business Communication by Asha Kaul / Prentice Hall
- Technical Communication: Principles and Practice by Meenakshi Raman & Sangeeta Sharma / Oxford

---

(Note: Each Session is of 2 hours)

GENERAL APTITUDE (32 hours)

Session 1:
- Number Systems
- Series & Cyclicity

Session 2:
- Average
- Percentage

Session 3:
- Ratio & Proportion

Session 4:
- Time & Work

Session 5:
- Time & Wages (Mandays)

Session 6:
- Probability

Session 7:
- Permutations & Combinations

Session 8:
- Profit & Loss

Session 9:
Session 10:
- Time, Speed & Distance
- Streams, Boats & Trains

Session 11:
- Mixtures & Alligations

Session 12:
- Puzzles

Session 13:
- Data Interpretation
- Syllogism
- Coding & Decoding

Session 14:
- Seating Arrangements

Session 15:
- Blood Relations
- Ages

Session 16:
- Clock
- Calendar
- Simple Interest & Compound Interest

EFFECTIVE COMMUNCIATION (28 hours)

Session 1:
Fundamentals of Communication
The Art of Communication

- Vocabulary, spelling and grammar
- Fluency, pronunciation, intonation and accent
- Idioms
- Synonyms & Antonyms

Practice Sessions:
Practise words, spelling, intonation and correct pronunciation
Practise idioms, synonyms & antonyms

Session 2:
Personality Development

- Greeting
- Etiquettes
- Body language
- Developing positive attitude
- Confidence building
- Questioning techniques

Practice Sessions:
Practise greeting, etiquettes and questioning

Session 3
English Grammar

- Nouns
- Pronouns
- Adjectives
- Articles
- Verbs
- Adverbs
- Prepositions
- Conjunctions

Practice Sessions:
Practise sentence making

Session 4:
English Grammar

- Active and passive voices
- Direct and indirect speeches

Practice Sessions:
Practise speaking in active & passive voices
Practise direct & indirect speaking

Session 5:
Correct usage of English
Common mistakes in English communication

Practice Sessions:
Practise general English communication

Session 6:
Listening Skills

- Importance of listening
- Techniques for effective listening
- Audio synthesis
  - Listening to audio clips
  - Question-answers based on the listened audio clips

Practice Sessions:
Practise audio synthesis

Session 7:
Reading Skills

- Comprehension
  - Techniques

Practice Sessions:
Comprehension exercises

Session 8:
Written Communication

- Essay writing
  - Characteristics of a good essay
  - Types of essays
  - Structure of an essay (introduction, main body, conclusion)
- Letter writing
  - Types of letters
  - Parts of a letter
- Official emailing
  - Structure and etiquettes of email writing
  - Tips to write an impressive email

Practice Sessions:
Essay writing
Letter writing
email writing

Session 9:
Public Speaking

- Speech design
- Informative speeches
- Speeches for special occasions (Introduction, Welcome, Felicitation, Thanks, etc)
- Extempore & impromptu speeches

Practice Sessions:
Conduct various types of speeches

Session 10:
Presentation Skills

- How to conduct effective and engaging presentations?
- Organisation & structure of presentation
- Design of slides in PPT
- Body language & voice

Practice Sessions:
Conduct presentations using PPT
Feedback of presentations

Session 11:
Group Discussions

- What is a GD?
- Skills assessed in GD
- Common mistakes
- Common GD topics

Practice Sessions:
Conduct practice GDs with video recording
Playing and analysis of GDs conducted

Session 12:
Personal Interviews

- Preparation for Interview
  - Qualities interviewers looking for
  - Getting ready for Interviews
  - Company research
  - Overall approach
  - Just before interview

Session 13:
Personal Interviews

- Introducing yourself
  - Importance of introduction
  - Structure of introduction

Practice Sessions:
Practise introduction
Analysis and feedback on introduction

Session 14:
Personal Interviews

- Facing job interviews
  - Confidence
  - Body language

    - Right mindset
- Tips for facing Interviews
    - What to do (and not do) during interviews?
    - Best practices and common mistakes of answering questions

Practice Sessions:
Practise common technical questions
Practise common HR/behavioral questions
Conduct mock interviews

Teaching Guidelines for

Software Project

Diploma in Advanced Computing (e-DAC)

September 2020

Duration: 100 hours

Objective: In addition to the specific subject knowledge, the Project module attempts to put into practice a number of things that the student has learned during the PG-DAC course, such as:

- Ability to work in a team
- Software development methodology
- Good programming practices
- Technical reporting and presentation.

Prerequisites: Completion of the basic modules on Programming, Data Structures, and Database to start Phase I of the Project.

Evaluation: Grading based on the combined marks obtained in the evaluations of both the phases of the project work.

Weightage: Phase I – 20%, Phase II – 80% (Mid Evaluation 20% + Final Evaluation 60%)

Software Project Schedule

Students in teams will be required to identify project topics in consultation with faculty members within the first three months of the course.

The Software Project module is divided in two phases.

Phase I – SRS & Design

Tasks: Project finalisation, requirements gathering, feasibility study.
Software design and project plan.

Deliverable: Software Requirement Specification (SRS).
Students will present the design and plan on the schema of the project.

Schedule: This phase will be executed along with the Software Development Methodologies sessions to enable better absorption of the concepts.

Phase II – Development & Testing

Tasks: Coding and testing of the software system/application to be developed.

Deliverables: Project report, functional software system/application.

Schedule: This final phase will be executed during the last month of the course. A mid evaluation at the middle of the project development, and a final evaluation at the end of the project will be done.