ocac

Technical Directorate of Information Technology Department

Govt. of Odisha

OCAC Training Centre

INTERNSHIP TRAINING PROGRAM

JAVA

About OCAC Training Centre

OCAC Training Centre is the state-of-the-art training centre established by OCAC the Designated Technical Directorate of Information Technology Department, Govt. of Odisha to develop the IT skills of learners and make them job-ready/ workforce ready with support from Odisha Knowledge Corporation Ltd. (OKCL).

OCAC Training Centre provide Summer Training & Internship for all the technical Graduates in the state of Odisha.

Here are some key benefits of the program that we believe will be of interest to students:

- ✓ **Learn from expert trainers with real-world experience in C++ / Python / Java.**
- ✓ **Work on a short-term project that showcases their technical skills.**
- ✓ **Get personalized guidance and feedback to help them excel.**
- ✓ **Receive a completion certificate from OCAC, a reputed government organization.**
- ✓ **Develop essential interview skills to ace technical and personal interviews.**
- ✓ **Guidance & opportunity for Placement**

Content

Internship Training programs

The OCAC Training Centre offers Internship Training programs to B-Tech and MCA students during the summer. These internship programs provide students with an opportunity to learn a programming language through assignment-based learning systems. The training program is designed to enhance students' practical skills and knowledge in their chosen programming language.

Here are some key features of the Internship Training program:

1. Programming Language Learning: Students will undergo classroom learning sessions where they will be taught the fundamentals and advanced concepts of a specific programming language. The focus is on developing a strong understanding of the language and its syntax.

2. Assignment-Based Learning Systems: The training program utilizes assignment-based learning systems, which means students will receive assignments related to the topics covered in the classroom. These assignments are designed to reinforce the theoretical concepts learned and encourage practical application.

3. Small Project Development: As part of the internship training, students will be required to work on a small project based on the learning outcomes. This project allows students to apply their knowledge and skills in a real-world scenario, fostering practical experience and problem-solving abilities.

4. Duration: The total duration of the Internship Training program is 60 hours. This includes 40 hours of classroom learning, where students receive theoretical instruction and hands-on practice, and an additional 20 hours dedicated to project development.

The Internship Training program aims to provide students with a comprehensive learning experience that combines theoretical knowledge with practical application. By completing assignments and working on a small project, students gain valuable skills and experience in their chosen programming language, preparing them for future career opportunities in the field of software development.

Course Curriculum for Internship Training in Java

Internship Training program in Java at the OCAC Training Centre, incorporating a Brush-up session on a core Java concept, 40 hours on Advanced Java, and 20 hours of project work using Java.

Duration: 70 hours (10 hours of Brush-up session + 40 hours of Advanced Java + 20 hours of project work)

Classroom Learning:

1. **Brush-up Session on Core Java (10 hours):**
    - Review of core Java concepts and syntax
    - Refreshing knowledge on essential topics such as data types, control flow, arrays, OOP concepts, exception handling, and file I/O.

Assignment:

1. Review of core Java concepts and syntax: a. Write a program to find the sum of the first 100 prime numbers. b. Implement a program that generates the Fibonacci series up to a given limit. c. Design a program to calculate the factorial of a number using recursion.

2. Refreshing knowledge on data types: a. Write a program to convert temperature from Celsius to Fahrenheit and vice versa. b. Implement a program that calculates the area and perimeter of different geometric shapes such as a circle, rectangle, and triangle. c. Design a program that determines if a given number is prime or not.

3. Control flow: a. Write a program to check if a given year is a leap year or not. b. Implement a program that prints the multiplication table for a given number up to a specific range. c. Design a program that finds the largest among three numbers using conditional statements.

4. Arrays: a. Write a program to find the second largest element in an array. b. Implement a program that sorts an array of integers in ascending order using the bubble sort algorithm. c. Design a program that calculates the average of all the elements in an array.

5. Object-oriented programming (OOP) concepts: a. Create a class representing a student with attributes like name, age, and grade. Implement methods to calculate the average grade and display student information. b. Implement a class hierarchy for different shapes (circle, square, triangle) with appropriate inheritance and polymorphism. c. Design a class representing a bank account with methods for deposit, withdrawal, and balance inquiry.

6. Exception handling: a. Write a program that reads input from a file and handles exceptions such as file not found or invalid data. b. Design a function that divides two

numbers and handles exceptions for division by zero. c. Create a program that performs input validation and throws custom exceptions for invalid user inputs.

7. File I/O operations: a. Write a program that reads data from a text file, manipulates the data, and writes the modified data to a new file. b. Design a program that reads a CSV file containing student records and performs operations like sorting and filtering based on user input. c. Create a program that reads binary data from a file, performs operations like encryption or compression, and writes the modified data to a new file.

2. **Advanced Java (40 hours):**

   - Java Generics and Collections
   - Multithreading and Concurrency
   - Networking and Socket Programming
   - Database Connectivity and JDBC
   - Java Servlets and JavaServer Pages (JSP)
   - Introduction to Java Frameworks (e.g., Spring, Hibernate)

Assignment:

1. Java Generics and Collections: a. Implement a generic stack data structure using an array or linked list. b. Design a program that sorts a collection of objects using a custom comparator. c. Create a generic method that finds the maximum element in a collection.
2. Multithreading and Concurrency: a. Write a program that simulates a multi-threaded scenario with producer and consumer threads sharing a bounded buffer. b. Implement a program that uses thread synchronization mechanisms like locks or semaphores to ensure thread safety. c. Design a concurrent program that performs parallel computations on a large data set using thread pooling.
3. Networking and Socket Programming: a. Create a client-server application that allows communication between multiple clients and a server using sockets. b. Design a program that performs network discovery by scanning a range of IP addresses and identifying active hosts. c. Implement a file transfer program using socket programming to send files between a client and a server.
4. Database Connectivity and JDBC: a. Write a program that connects to a database, retrieves data from a table, and performs CRUD operations using JDBC. b. Design a program that uses prepared statements to execute parameterized queries and prevent SQL injection attacks. c. Implement a database-backed application that handles transactions and performs batch updates using JDBC.

5. Java Servlets and JavaServer Pages (JSP): a. Create a simple web application using servlets and JSP to handle user registration and login functionality. b. Design a program that utilizes servlet filters for authentication and authorization of incoming requests. c. Implement a dynamic web application that displays real-time data using servlets and JSP.
6. Introduction to Java Frameworks (e.g., Spring, Hibernate): a. Build a Spring MVC application that performs CRUD operations on a database using Spring Data JPA. b. Design a program that utilizes Spring AOP to implement logging or security aspects in a Java application. c. Implement a Hibernate-based application that maps Java objects to database tables and performs ORM operations.

Project Work (20 hours): During the project phase, students will work on a Java project that integrates the advanced Java concepts covered in the classroom. The project work will involve designing and implementing a practical application, utilizing frameworks and technologies such as servlets, JSP, JDBC, or other relevant components. The specific project topic will be assigned by the instructors based on the learning objectives and the complexity suitable for the internship duration.

The curriculum includes a Brush-up session to refresh core Java concepts before diving into Advanced Java topics. This helps ensure a solid foundation and smooth transition to more advanced concepts. The Advanced Java segment covers essential topics that extend beyond the basics, providing students with a deeper understanding of Java programming and its application in real-world scenarios. The project work component allows students to apply their knowledge and skills to a meaningful Java project, further enhancing their practical experience.

Brush-up Session on Core Java

A detailed syllabus for the Brush-up Session on Core Java, which spans 10 hours of instruction:

1. Overview of Java:
    - Introduction to the Java programming language
    - Features of Java
    - Java development environment setup (IDE)
2. Java Basics:
    - Variables and data types

- Operators and expressions
- Control flow statements (if, switch, loops)

3. Arrays and Strings:
    - Declaring and initializing arrays
    - Array operations (traversing, sorting, searching)
    - String manipulation and methods
4. Object-Oriented Programming (OOP) Concepts:
    - Classes and objects
    - Encapsulation, inheritance, and polymorphism
    - Constructors and method overloading
5. Exception Handling:
    - Understanding exceptions and errors
    - Handling exceptions using try-catch blocks
    - Throwing and creating custom exceptions
6. Input and Output Handling:
    - File I/O operations (reading, writing, appending)
    - Working with streams and readers/writers
    - Handling file exceptions
7. Java Collections Framework:
    - Introduction to collections (ArrayList, LinkedList, HashMap)
    - Iterating and manipulating collections
    - Sorting and searching collections
8. Generics:
    - Generic classes and methods
    - Type inference and bounded types
    - Working with generic collections
9. Multithreading Basics:
    - Introduction to threads and concurrency
    - Creating and managing threads

- Thread synchronization and communication

10. Recap and Practice:

    - Review of key concepts and topics covered
    - Practical exercises and coding challenges to reinforce learning

The Brush-up Session on Core Java is designed to refresh students' knowledge and understanding of essential concepts and syntax in Java programming. The syllabus covers a wide range of topics, including variables and control flow, object-oriented programming, exception handling, file I/O, collections, generics, and multithreading basics. Through this session, students will revisit and reinforce their understanding of core Java, preparing them for the more advanced topics to follow in the Advanced Java segment.

<u>Syllabus for the Advanced Java</u>

A detailed syllabus for the Advanced Java segment, which spans 40 hours of instruction:

1. Java Generics and Collections:
    - Introduction to generics and type parameters
    - Generic classes, methods, and interfaces
    - Type inference and wildcard usage
    - Collections framework enhancements with generics
2. Multithreading and Concurrency:
    - Understanding threads and thread lifecycle
    - Synchronization and locks
    - Thread safety and race conditions
    - Concurrent collections and utilities
    - Executors and thread pools
    - Thread communication and synchronization mechanisms
3. Networking and Socket Programming:
    - Basics of networking protocols (TCP/IP, UDP)
    - Socket programming fundamentals
    - Creating client-server applications
    - Handling network I/O and communication
    - Working with HTTP and URL connections
4. Database Connectivity and JDBC:
    - Introduction to JDBC (Java Database Connectivity)
    - Establishing database connections
    - Executing SQL queries and updates
    - Retrieving and manipulating result sets
    - PreparedStatement and CallableStatement usage
    - Transaction management and error handling
5. Java Servlets and JavaServer Pages (JSP):
    - Introduction to web applications and the Servlet API

- Servlet lifecycle and request handling
- Building dynamic web pages with JSP
- JSP directives, expressions, and scriptlets
- JavaBean integration with JSP
- Session management and filters

6. Introduction to Java Frameworks (e.g., Spring, Hibernate):
    - Overview of popular Java frameworks
    - Introduction to the Spring Framework and its core concepts (dependency injection, inversion of control)
    - Introduction to Hibernate ORM (Object-Relational Mapping)
    - Integrating Spring and Hibernate
    - Hands-on exercises and examples using the frameworks

The Advanced Java syllabus covers crucial topics in Java programming that go beyond the core concepts. Students will delve into Java Generics and Collections, enabling them to write more flexible and reusable code. The Multithreading and Concurrency module focuses on developing concurrent applications, ensuring thread safety, and utilizing advanced synchronization mechanisms. Networking and Socket Programming provide insights into communication over networks. The Database Connectivity and JDBC section equips students with skills to interact with databases using Java. Java Servlets and JSP cover web application development, and the Introduction to Java Frameworks introduces popular frameworks such as Spring and Hibernate.

It's important to note that the duration and depth of coverage for each topic may vary based on the Training Centre's curriculum and the pace of instruction. The syllabus aims to provide students with a comprehensive understanding of advanced Java concepts and tools commonly used in real-world applications.

Projects Using Java

Details of use-cases, technologies/tools/APIs required apart from Java for each of the suggested projects:

Project	Module-wise Use-Cases	Technologies/Tools/APIs
Online Quiz Application	- User registration and login	HTML, CSS, JavaScript, Java Servlets, JDBC
	- Quiz creation and management	
	- Quiz question display and submission	
	- Scoring and result calculation	
File Transfer Application	- Establishing client-server connection	Sockets, Networking
	- File selection and transfer	
	- Progress tracking	
Social Media Analytics Tool	- Data retrieval from social media APIs	API integration, Data analytics tools
	- Data preprocessing and analysis	
	- Generating insights and visualizations	
Employee Management System	- Employee record management	Database (MySQL, Oracle, etc.), Java Servlets, JDBC
	- Employee information search and retrieval	
	- Adding, updating, and deleting employee records	
Weather Forecast Application	- Retrieving weather data from API	API for weather data
	- Displaying current weather conditions and forecasts	
Online Voting System	- User registration and login	Database (MySQL, Oracle, etc.), Java Servlets, JDBC
	- Creating and managing voting categories and options	

	- Casting votes and calculating results	
Online Bookstore	- Browsing and searching for books	HTML, CSS, JavaScript, Java Servlets, JDBC
	- Adding books to the cart and placing orders	
	- Payment processing and order management	
Recipe Finder Application	- Searching recipes based on ingredients or cuisines	API for recipe data
	- Displaying recipe details and instructions	
Stock Market Analysis Tool	- Retrieving stock market data from API	API for stock market data
	- Analyzing stock trends and performance	
	- Generating charts and indicators	
Movie Recommendation System	- User registration and login	Database (MySQL, Oracle, etc.), Java Servlets, JDBC
	- Gathering user preferences and ratings	
	- Recommending movies based on user preferences	
Expense Tracker	- Adding and categorizing expenses	Database (MySQL, Oracle, etc.), Java Servlets, JDBC
	- Generating expense reports and summaries	
	- Setting budget limits and tracking spending	
Blogging Platform	- User registration and login	HTML, CSS, JavaScript, Java Servlets, JDBC
	- Creating and publishing blog posts	

	- Commenting on blog posts and interacting with other bloggers	
Event Management System	- Creating and managing events	Database (MySQL, Oracle, etc.), Java Servlets, JDBC
	- Sending event invitations and managing registrations	
	- Generating attendee lists and managing event details	
Online Auction System	- Posting items for auction and bidding	Database (MySQL, Oracle, etc.), Java Servlets, JDBC
	- Managing auction processes and time limits	
	- Tracking bids and determining auction winners	
Music Streaming Application	- Searching and playing songs	API for music data
	- Creating playlists and managing favorites	
	- Displaying song information and album covers	
Online Job Portal	- Job posting and searching	Database (MySQL, Oracle, etc.), Java Servlets, JDBC
	- Resume submission and application management	
	- Employer shortlisting and candidate selection	
Online Banking System	- Account management and transactions	Database (MySQL, Oracle, etc.), Java Servlets, JDBC
	- Fund transfers and transaction history	
	- Bill payments and account statements	
Online Ticket Booking System	- Event/movie selection and seat booking	Database (MySQL, Oracle, etc.), Java Servlets, JDBC

	- Payment processing and ticket generation	
	- Ticket cancellation and refund management	
Health Monitoring System	- Collecting health data from wearable devices	Wearable devices, API for health data
	- Data analysis and generating health insights	
	- Setting health goals and monitoring progress	
Travel Planner	- Destination suggestions and itinerary planning	API for travel data
	- Booking accommodations and attractions	
	- Providing travel tips and local information	

This table provides a breakdown of module-wise use-cases for each project, as well as the additional technologies, tools, and APIs that may be required apart from Java. The technologies, tools, and APIs vary depending on the specific functionalities and integrations required for each project. Please note that the table is not exhaustive, and additional technologies or tools may be needed based on the project's complexity and specific implementation requirements.

Software Project Development Process

To successfully complete the above projects within the given time frame and learn about software project development, students can follow a step-by-step process based on the Software Development Life Cycle (SDLC), incorporating various design concepts. Here's an extended version of the process:

1. Project Understanding and Planning:
    - Thoroughly understand the project requirements and objectives.
    - Identify the stakeholders and their needs.
    - Define project scope, constraints, and deliverables.
    - Create a project plan with specific milestones and deadlines.
2. Requirement Gathering and Analysis:
    - Conduct detailed requirement gathering sessions with stakeholders.
    - Document user stories, use cases, and functional requirements.
    - Use techniques like interviews, surveys, and brainstorming to gather requirements.
    - Analyze the gathered requirements and create a requirement specification document.
3. Design Phase:
    - Use the gathered requirements to design the system architecture.
    - Design the database schema using Entity-Relationship (ER) diagrams.
    - Create Data Flow Diagrams (DFDs) to illustrate data movement and processing within the system.
    - Design user interfaces using wireframes and mock-ups.
    - Apply user interface design principles to ensure usability and accessibility.
4. Development Iterations:
    - Break down the project into smaller tasks or user stories.

- Prioritize and tackle the tasks based on their importance and dependencies.
- Follow an iterative development approach, implementing one feature or module at a time.
- Write clean, modular, and well-commented code using best practices.

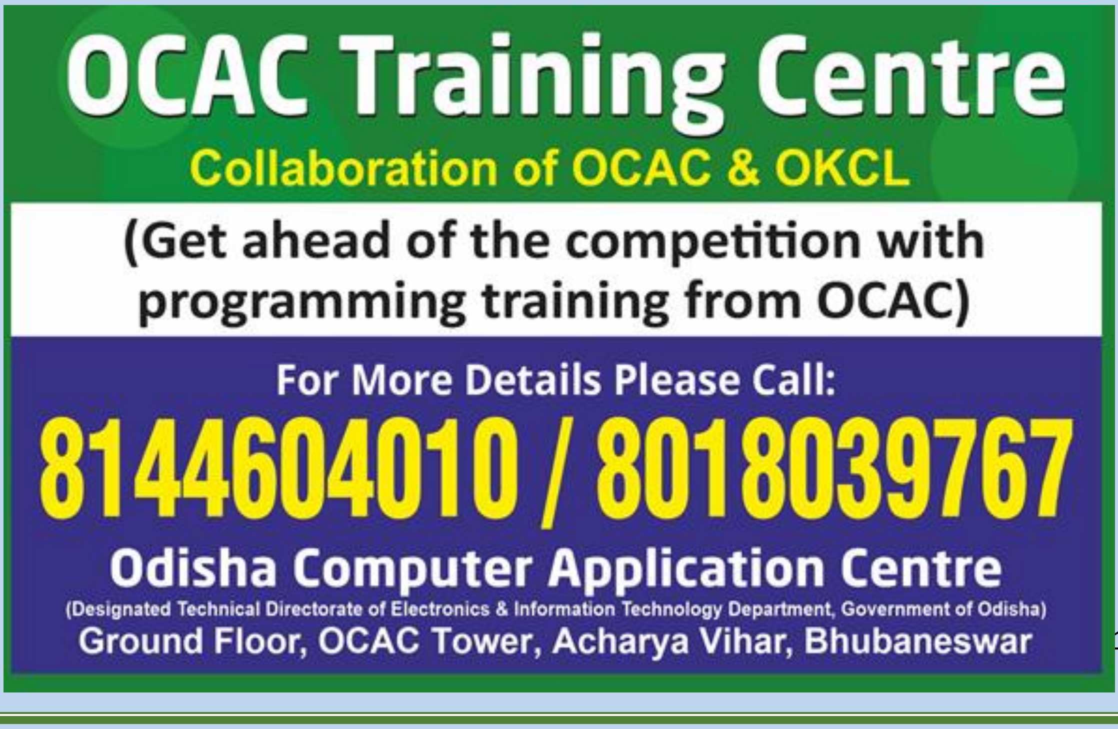
5. Testing and Debugging:
    - Conduct various types of testing, such as unit testing, integration testing, and system testing.
    - Write test cases to verify the functionality and identify any defects.
    - Perform regression testing to ensure that new changes do not impact existing features.
    - Debug and fix any issues or bugs that arise during testing.
6. Integration and Deployment:
    - Integrate the different components or modules to ensure they work together seamlessly.
    - Deploy the application to a development or testing environment.
    - Perform system-level testing and user acceptance testing.
    - Prepare the application for production deployment.
7. Documentation:
    - Document the project, including its architecture, design, and usage instructions.
    - Create user manuals, technical documentation, and system documentation.
    - Document any APIs, libraries, or tools used in the project.
    - Include installation and configuration instructions for the application.
8. Presentation and Demonstration:

- Prepare a presentation or demo to showcase the completed project.
- Explain the project's features, functionality, and implementation details.
- Use diagrams, such as ER diagrams and DFDs, to illustrate the system design and data flow.
- Discuss the user interface design choices and how they enhance user experience.

9. Reflection and Learning:
    - Reflect on the project development process and identify areas for improvement.
    - Evaluate the challenges faced, lessons learned, and skills acquired.
    - Consider feedback from stakeholders and users for future enhancements.
    - Document your learnings and make note of any future improvements to the project.

By following this comprehensive process, incorporating the SDLC stages and design concepts like use case design, ER diagrams, data flow diagrams, and user interface design, students can effectively complete the projects within the given time frame while gaining valuable experience in software project development.

Contact



OCAC Training Centre

Collaboration of OCAC & OKCL

(Get ahead of the competition with programming training from OCAC)

For More Details Please Call:

8144604010 / 8018039767

Odisha Computer Application Centre

(Designated Technical Directorate of Electronics & Information Technology Department, Government of Odisha)

Ground Floor, OCAC Tower, Acharya Vihar, Bhubaneswar