AMITY UNIVERSITY PATNA

SUMMER TRAINING REPORT

On

DATA STRUCTURES AND ALGORITHMS IN JAVA

Organized by Amity University Patna
In Collaboration with Coding Blocks Pvt. Ltd.

Training Duration: From: 23rd June 2025

To: 6th August 2025

Submitted By: Udipta Kumar
Enrollment No.: A5605222023
Course: B.Tech in Computer Science and Engineering
Department: ASET
Under the Guidance of : Harsh Tripathi(Trainer from Coding Blocks Pvt. Ltd.)
Submitted To: Dr. Abhishek Anand

CERTIFICATE

This is to certify that Mr. Udipta Kumar, a student of B.Tech in Computer Science and Engineering, bearing Enrollment No.: A5605222023, of Amity School of Engineering and Technology (ASET), Amity University Patna, has successfully completed the Summer Training Program on
"Data Structures and Algorithms in Java".

The training was organized by Amity University Patna in collaboration with Coding Blocks Pvt . Ltd. and was conducted from 23rd June 2025 to 6th August 2025 (a total duration of 45 days).

During this period, he has shown sincere effort, active participation, and dedication towards learning and practical implementation of various concepts covered in the training program.

We appreciate his hard work and wish him success in all his future endeavors

Faculty Coordinator (Signature) External Trainer (Signature)

Head of Department (Signature & Seal)

ACKNOWLEDGEMENT

I would like to express my sincere gratitude to **Amity University Patna** for providing me with the opportunity to attend the **Summer Training Program** on ***Data Structures and Algorithms in Java***, organized in collaboration with **Coding Blocks Pvt. Ltd.**

I am especially thankful to the **faculty members and coordinators** of the **Amity School of Engineering and Technology (ASET)** for their continuous guidance, encouragement, and support throughout the training period. Their motivation helped me to enhance my technical skills and gain a deeper understanding of programming concepts.

My heartfelt thanks also go to the trainer **Mr. Harsh Tripathi (from Code Basics Pvt. Ltd.)**, who explained every concept with great clarity and provided valuable insights into problem-solving techniques using Java. Their teaching made complex topics much easier to understand and apply in practice.

I would also like to thank my **friends and classmates** for their cooperation and teamwork during the training sessions and assignments. Lastly, I express my gratitude to my **parents** for their constant encouragement and moral support throughout this journey.

This training has been a great learning experience that helped me to strengthen my basics in Java and develop problem-solving skills which will be very useful in my future career.

INDEX

COMPANY PROFILE

Coding Blocks Pvt. Ltd. is a leading software training and education company that focuses on bridging the gap between academic learning and industry-level programming skills. Established with the aim of improving the coding culture among students, Coding Blocks provides high-quality training in computer programming, data structures, algorithms, web development, and other software technologies.

The company's main objective is to make technical learning simple, practical, and effective for students who wish to build a strong foundation in programming. Coding Blocks offers both online and offline courses, workshops, and bootcamps designed to help learners master real-world problem-solving and prepare for technical interviews in top IT companies.

The trainers at Coding Blocks are experienced professionals and industry experts who bring hands-on experience and in-depth knowledge to the classroom. Their teaching approach emphasizes *learning by doing*, where students work on numerous coding exercises, assignments, and projects. This approach helps learners apply theoretical concepts to real-life problems and understand programming logic in depth.

During the summer training organized by Amity University Patna, Coding Blocks Pvt. Ltd. conducted sessions on **Data Structures and Algorithms using Java**, where students were taught the fundamentals of programming, various data structures, algorithmic problem-solving techniques, and project-based applications.

Coding Blocks is well known for its structured curriculum, interactive teaching style, and focus on practical implementation. It has trained thousands of students across India, helping them become confident coders and preparing them for competitive programming and software development careers.

With its mission to make every student an efficient and logical programmer, **Coding Blocks Pvt. Ltd.** continues to be one of the most reputed and trusted names in the field of computer science education and professional training

INTRODUCTION

The **Summer Training Program** is an important part of the B.Tech curriculum at **Amity University Patna**. It provides students an opportunity to gain practical knowledge and industrial exposure beyond classroom learning. As part of this program, I attended a **45-day summer training on "Data Structures and Algorithms in Java"**, organized by **Amity University Patna** in collaboration with **Coding Blocks Pvt. Ltd.**, from (**23rd June 2025 to 6th August 2025)**.

The main objective of this training was to help students strengthen their programming skills and understand the core concepts of **Data Structures and Algorithms (DSA)** using the **Java programming language**. DSA forms the backbone of efficient coding and problem-solving, which are essential for software development and competitive programming.

During the training period, the sessions covered a wide range of topics such as arrays, linked lists, stacks, queues, recursion, trees, graphs, sorting, and searching algorithms. The focus was not only on theoretical understanding but also on implementing these concepts practically through coding exercises and assignments.

The training also included multiple **hands-on coding sessions**, **daily practice problems** which helped in building logical thinking and writing optimized code. The instructors had explained every topic in a clear and structured manner along with dry runs and algorithms, ensuring that each concept was understood and applied effectively.

Overall, this summer training served as a valuable learning experience, improving my technical knowledge, problem-solving ability, and confidence in programming. It has prepared me for future academic projects, technical interviews, and real-world software development challenges

LEARNING OUTCOME

The **45-day Summer Training Program** on **Data Structures and Algorithms in Java** proved to be a highly beneficial learning experience. It helped me to improve my programming skills and understand how data structures and algorithms form the base of efficient coding and problem-solving. Throughout the training, I learned to apply theoretical concepts into practical coding exercises, which strengthened my logical thinking and analytical ability.

Below are the key learning outcomes from this training:

1. **Strong Understanding of Core Java Concepts:**
   I gained a clear understanding of Java fundamentals such as data types, control statements, loops, functions, arrays, and object-oriented programming concepts like classes, objects, inheritance, polymorphism, and encapsulation.

2. **Practical Knowledge of Data Structures:**
   I learned how to implement and use various data structures such as arrays, linked lists, stacks, queues, hash maps, trees, heaps, and graphs. I also understood their real-life applications and time complexities.

3. **Algorithmic Thinking:**
   The training improved my ability to think algorithmically. I practiced different algorithms such as sorting (bubble, insertion, merge, quick), searching (linear and binary), and recursion-based solutions.

4. **Problem-Solving Skills:**
   By solving daily coding assignments and challenges, I became more confident in approaching problems logically and breaking them into smaller sub-tasks for efficient solutions.

5. **Improved Coding Efficiency:**
   I learned how to write clean, readable, and optimized code in Java using proper syntax, indentation, and meaningful variable naming.

6. **Hands-on Project Experience:**
   Working on small practice projects helped me understand the process of applying multiple concepts together to create functional and working programs.

7. **Use of Tools and IDEs:**
   I got familiar with different development tools like IntelliJ IDEA and Eclipse, which made the coding process more convenient and professional.

8. **Teamwork and Discipline:**
   The structured daily schedule, along with discussions and interactions with peers and mentors, improved my teamwork, discipline, and time management skills.

Overall, this training has enhanced my confidence in Java programming and built a solid foundation for advanced subjects like competitive programming, software development, and data analysis

MODULES COVERED

The **Summer Training on Data Structures and Algorithms in Java** was divided into several well-structured modules that covered both the theoretical and practical aspects of Java programming and DSA concepts. The sessions included explanations, live coding demonstrations, and daily practice exercises.

Below are the major modules covered during the training:

Module 1: Introduction to Java Programming

- Overview of Java and its features
- Understanding JDK, JRE, and JVM
- Setting up the development environment
- Writing and executing simple Java programs
- Data types, variables, operators, and expressions
- Input/output operations in Java

Module 2: Control Statements and Functions

- Conditional statements (if, else-if, switch)
- Looping constructs (for, while, do-while)
- Break and continue statements
- Methods in Java and method overloading
- Recursion and its applications

Module 3: Object-Oriented Programming in Java

- Classes and objects

- Constructors and types of constructors
- Inheritance and types of inheritance
- Polymorphism (compile-time and runtime)
- Encapsulation and abstraction
- Access modifiers and static members

Module 4: Arrays and Strings

- One-dimensional and two-dimensional arrays
- Array operations (traversal, insertion, deletion)
- String handling and manipulation
- Common string algorithms and methods

Module 5: Data Structures in Java

- Stack and its operations (push, pop, peek)
- Queue and types (simple, circular, priority queue)
- Linked List (singly, doubly, and circular)
- HashMap, HashSet, and TreeMap
- Applications and performance analysis

Module 6: Algorithms and Problem Solving

- Searching algorithms (Linear and Binary Search)
- Sorting algorithms (Bubble, Insertion, Selection, Merge, Quick Sort)
- Recursion-based problems
- Divide and Conquer approach
- Time and space complexity analysis

Module 7: Advanced Data Structures

- Trees (Binary Tree, Binary Search Tree, AVL Tree)
- Graphs (Representation, Traversal – BFS, DFS)
- Heap and Priority Queue
- Dynamic Programming (introduction and basic problems)

Module 8: Hands-on Coding

- Daily coding assignments and practice problems
- Implementation of real-world DSA-based problems
- Code review and optimization techniques

Each module was followed by practical assignments and interactive discussions which made the concepts clearer and easier to apply. This modular structure ensured a step-by-step understanding of both **Java programming** and **Data Structures and Algorithms** in a comprehensive way.

TOOLS AND TECHNOLOGIES USED

During the **45-day Summer Training on DSA in Java**, several tools, software platforms, and technologies were used to enhance the learning process and make coding practice more efficient.

Below is the list of major tools and technologies used during the training:

---

1. Programming Language: Java

- The entire training was focused on the **Java programming language**.
- Concepts of both **Core Java** and **Object-Oriented Programming (OOPs)** were applied for solving DSA problems.

---

2. Integrated Development Environments (IDEs)

- **Visual Studio Code:**
  Used as the main coding environment for writing, running, and debugging Java programs. It provided code suggestions, easy error detection, and smooth compilation.
- **Eclipse IDE:**
  Also introduced during the training for understanding another Java development platform. It helped in learning project setup and organization.

---

3. Online Coding Platforms

- **Coding Blocks Learning Portal:**
  The official training platform provided by **Coding Blocks** where daily topics, assignments, and test series were uploaded.
- **LeetCode:**
  Used for practicing coding problems related to arrays, strings, recursion, and sorting. The platform helped us to improve logic-building and problem-solving skills.

---

5. Version Control and Documentation

- **Git and GitHub (Basic Introduction):**
  Basic concepts of version control were explained, including repository creation, code upload, and collaboration using GitHub.
- **Microsoft Word and Excel:**
  Used for documentation, tracking progress, and preparing practice sheets and reports.

---

6. Hardware and Operating System

- **Laptop/PC with Windows OS**
  Used for all development activities. Java JDK and IDEs were installed and configured for running programs smoothly.

These tools and technologies collectively created a strong learning environment and helped in understanding both the **theoretical and practical** aspects of DSA in Java. Working with these platforms also gave me an experience of real-world software development practices

ASSIGNMENTS

During the **Summer Training Program on DSA in Java**, daily assignments and coding exercises were given. These assignments were designed to strengthen the understanding of concepts taught in class and to encourage regular coding practice.

Each assignment was based on the module covered during the session and aimed to help us apply the theoretical concepts practically. The assignments gradually increased in difficulty, starting from simple Java programs and moving toward complex data structure and algorithm problems.

Below is an overview of the types of assignments and exercises completed during the training:

---

1. Basic Java Assignments

- Writing simple Java programs to print patterns, calculate sums, and perform arithmetic operations.
- Programs using loops, conditional statements, and functions.
- Practice on input/output operations and type casting.

Examples:

- Write a Java program to check whether a number is even or odd.
- Program to find factorial of a number using recursion.
- Write a program to print Fibonacci series up to n terms.

---

2. Object-Oriented Programming Assignments

- Implementation of classes and objects.
- Programs on constructors, method overloading, and inheritance.
- Practical understanding of polymorphism, abstraction, and encapsulation.

Examples:

- Create a class BankAccount with deposit and withdrawal methods.

- Implement multiple inheritance using interfaces.
- Write a Java program to demonstrate method overriding.

---

3. Data Structures Assignments

- Implementation of basic data structures such as arrays, linked lists, stacks, and queues.
- Performing insertion, deletion, and traversal operations.

Examples:

- Implement a stack using arrays.
- Create a program to reverse a linked list.
- Implement a queue and perform enqueue and dequeue operations.

---

4. Algorithm-Based Assignments

- Solving problems using searching and sorting algorithms.
- Understanding time and space complexity through different implementations.

Examples:

- Write a program for binary search in a sorted array.
- Implement bubble sort and quick sort algorithms.
- Find the second largest element in an array.

---

5. Recursion and Problem-Solving Assignments

- Solving mathematical and logical problems using recursion.
- Understanding base and recursive cases for different functions.

Examples:

- Program to calculate the sum of digits of a number using recursion.
- Tower of Hanoi , sudoku and some more back tracking problems .

---

6. Advanced DSA Assignments

- Tree and graph-based problems.
- Practice on real-world problems using data structures like hash maps and priority queues.

Examples:

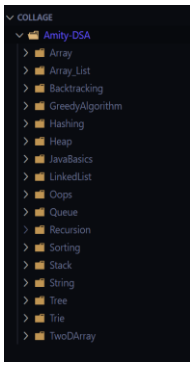
- Implement binary tree traversal (inorder, preorder, postorder).
- Write a program to detect a cycle in a graph using DFS.

These assignments helped me gain confidence in Java programming and improved my problem-solving approach. By completing more than **100+ coding exercises** and **practice problems**, I was able to strengthen my understanding of algorithms, time complexity, and efficient coding techniques

PROJECT SCREENSHOTS

During the training, many coding problems were solved. Some of the important ones, along with their outputs, are shown below.

1.Directory



2(a). N-Queen Problem

2(b).Input by user

2(c).Output

3(a).Bubble Sort

3(b).Output

4(a). FindNextGreatestElementUsingStack

4(b).Output

CHALLENGES FACED AND SOLUTIONS IMPLEMENTED

During the Training I faced several challenges related to understanding complex topics, debugging programs, and managing time effectively. With regular practice, trainer guidance, and self-study, I was able to overcome these difficulties and grow both technically and personally.

---

1. Understanding Complex Concepts

At first, I found it difficult to understand advanced data structures like trees, graphs, and hash maps. To overcome this, I watched additional tutorials, practiced small code examples to understand how data moves through these structures.

---

2. Debugging Errors

Frequent logical and runtime errors in my code slowed my progress. I learned to use debugging tools in **IntelliJ IDEA** and **Eclipse** and how to use ai for fixing them.

---

3. Managing Time

Completing daily assignments on time was challenging, especially in complex topics like recursion and dynamic programming. I developed a daily study routine and practiced extra coding problems after class, which improved my time management and consistency.

---

4. Applying Theory to Practice

Initially, I could understand concepts but struggled to apply them. By solving problems on **LeetCode** and companies website , I connected theory with practical coding, improving my analytical and problem-solving skills.

---

5. Handling Recursion

Recursion and algorithm complexity were confusing at first. I started using dry runs to trace recursive calls, which helped me understand base cases and write more optimized functions.