Cognizant Digital Nurture 4.0 – Week 1:

# M. Aditya Naidu 4992389

# Topic: Design Patterns, Data Structures & Algorithms

## 1. Design Patterns

Design patterns offer reusable solutions to common design problems.

Types of Design Patterns:

* Creational – Singleton, Factory, Builder
* Structural – Adapter, Decorator, Composite
* Behavioral – Observer, Strategy, Command

Example: Singleton Pattern in Java

public class Singleton {  
 private static Singleton instance;  
 private Singleton() {}  
 public static Singleton getInstance() {  
 if (instance == null) {  
 instance = new Singleton();  
 }  
 return instance;  
 }  
}

Output: Singleton instance created only once.

## 2. Data Structures

Data structures help in organizing and storing data. Choosing the right data structure can significantly affect performance.

|  |  |  |
| --- | --- | --- |
| Type | Examples | Use Cases |
| Linear | Array, Linked List, Stack, Queue | Memory storage, function calls |
| Non-Linear | Tree, Graph | Hierarchies, Networks |
| Hash-based | HashMap, HashSet | Fast lookups, uniqueness constraints |

Example: Stack using Java

import java.util.Stack;  
  
public class StackExample {  
 public static void main(String[] args) {  
 Stack<Integer> s = new Stack<>();  
 s.push(10);  
 s.push(20);  
 System.out.println(s.pop());  
 }  
}

Output: 20

## 3. Algorithms

=> Algorithms define a step-by-step method to solve problems.

Common types:

* Searching – Linear Search, Binary Search
* Sorting – Bubble Sort, Merge Sort, Quick Sort
* Graph – BFS, DFS
* Dynamic Programming – Knapsack, Fibonacci

Example: Binary Search in Java

public class BinarySearch {  
 public static int search(int[] arr, int target) {  
 int low = 0, high = arr.length - 1;  
 while (low <= high) {  
 int mid = (low + high) / 2;  
 if (arr[mid] == target) return mid;  
 else if (arr[mid] < target) low = mid + 1;  
 else high = mid - 1;  
 }  
 return -1;  
 }  
}

Output: Index of target element or -1