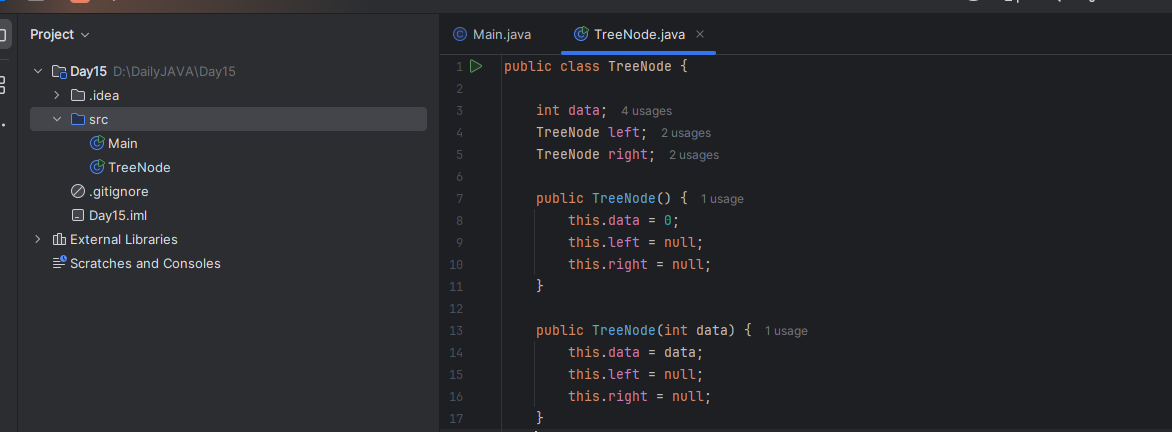
**ID: bvarshsp**

**Name: Varsha SP**

**Day 15 – 3nd July 2025**

**Task 1:**

Create a node for a tree and include a constructor (empty)

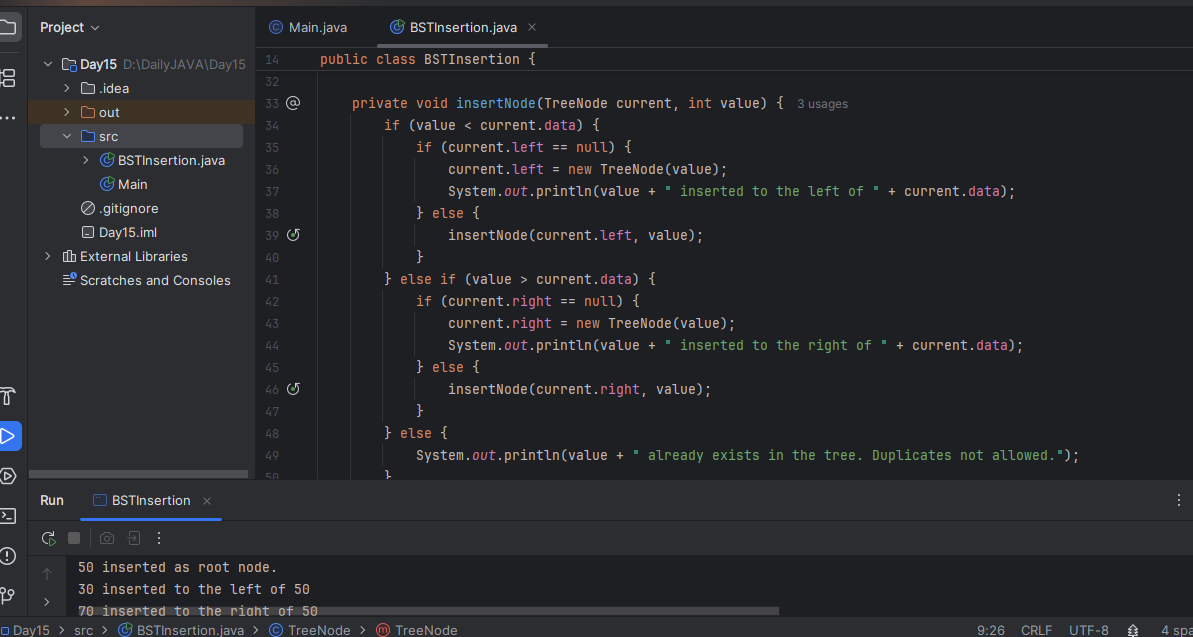


**Task 2:**

Create a class named Binary Search tree in which you have 2 insert operations

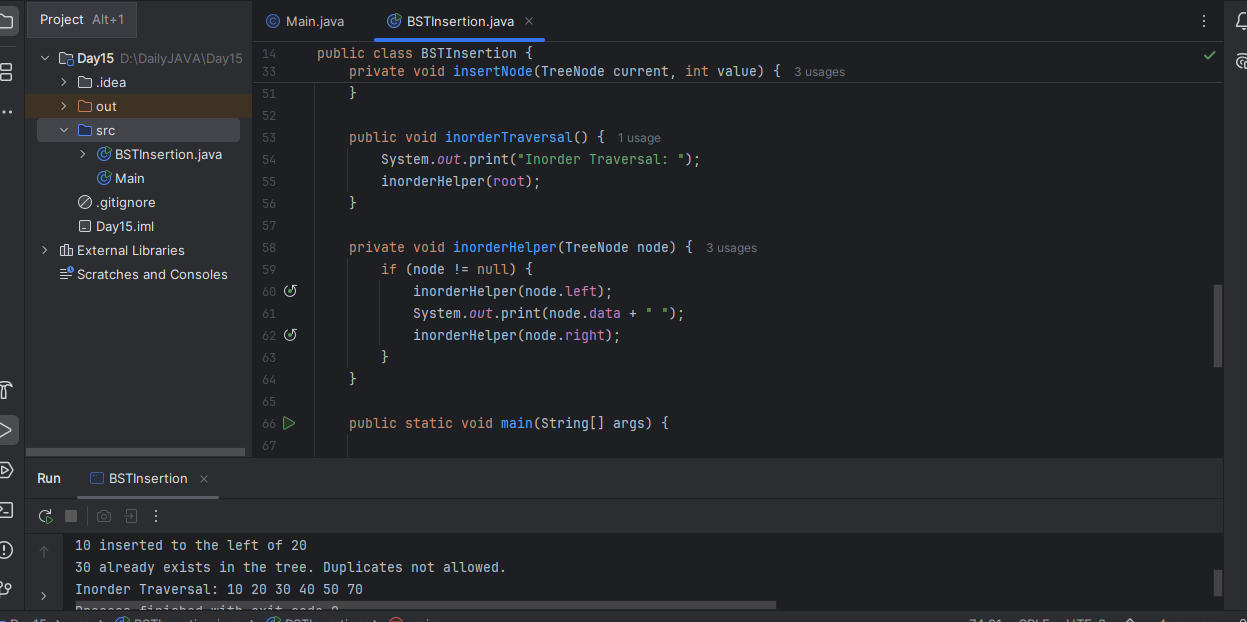
1 insert —----> for inserting if the tree is empty

1 insert —----> for inserting if the tree is 1 or more nodes



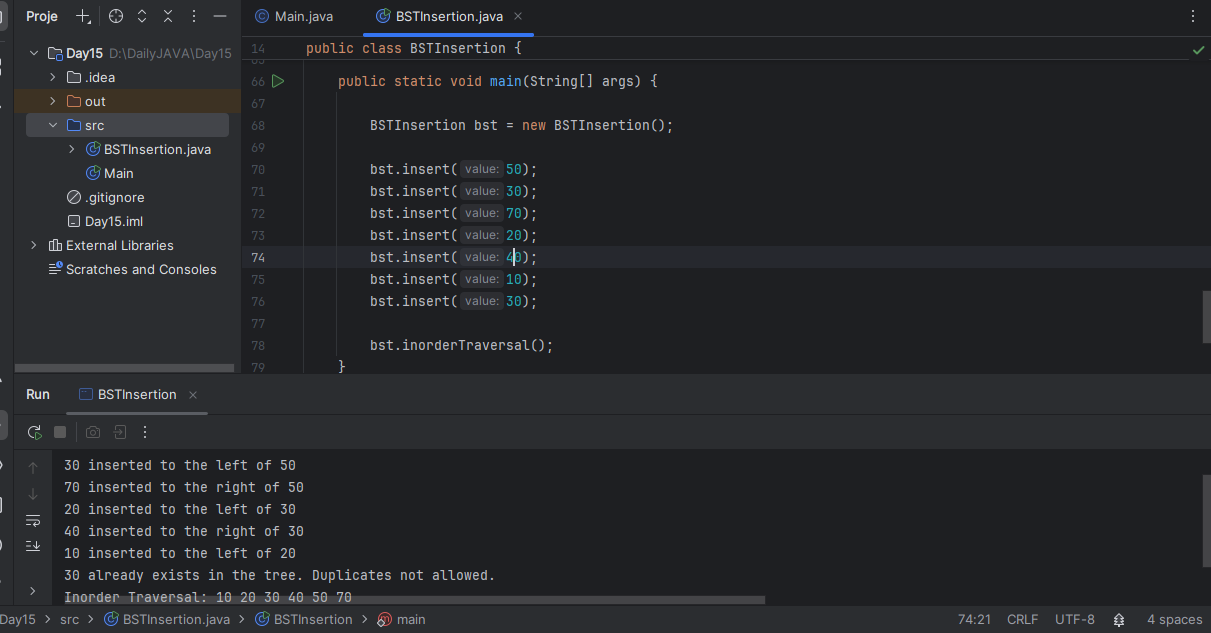
**Task 3:**

In-order travel of the above code snippets from task 1 and Task 2



**Task 4:**

Create a main method Task 1, 2 and 3



**Task5:** Applications of Trees:

Answer:

**Hierarchical Data Representation**

* Example: File system (folders inside folders)

**Database Indexing**

* Example: B-Trees and B+ Trees are used to organize and search databases efficiently.

**Routing and Network Structure**

* Example: Internet routing paths use trees to manage decisions.

**Search Operations**

* Example: Binary Search Trees (BST) allow fast searching, insertion, and deletion.

**Expression Evaluation**

* Example: Expression trees represent arithmetic expressions like ((3 + 2) \* 5).

**Artificial Intelligence**

* Example: Decision Trees are used in machine learning for classification and predictions.

**Compilers**

* Example: Abstract Syntax Trees (AST) are used to represent program structure during parsing.

**Autocomplete & Spell Check**

* Example: Tries (Prefix Trees) are used in dictionaries and search bars.

**HTML/XML Document Structure**

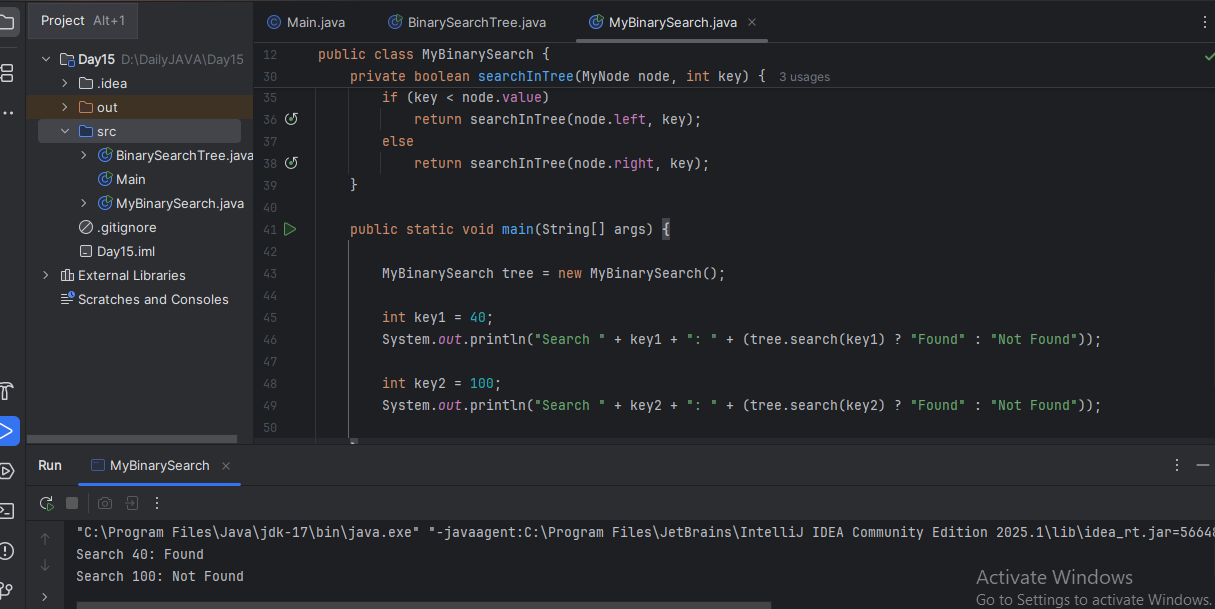
* Example: Browsers use DOM trees to render nested HTML elements.

**File Explorer on Your Computer**

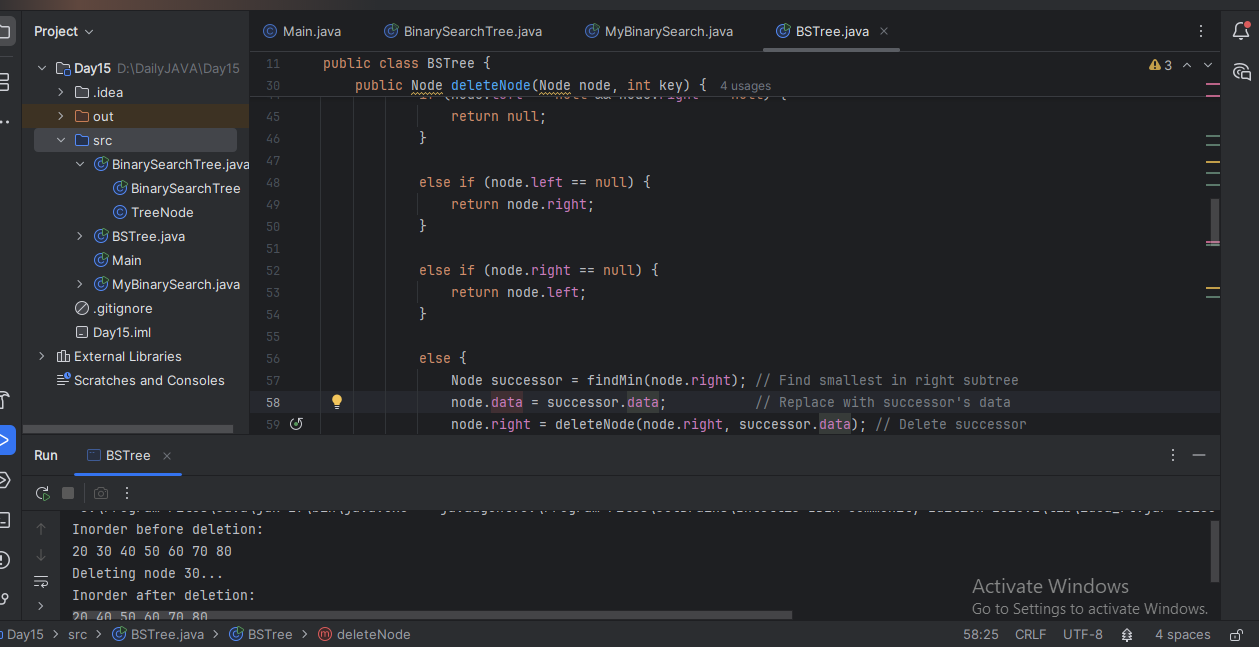
* The folder → subfolder → file structure is a **tree**.
* Root folder → Subfolders → Files (like a tree with branches and leaves).

**Task 6:**

Create a binary search operation on tree



**Task7:** Delete operation on BST



**Task 8:**

Types of binary trees:

Answer:

1. Rooted Binary Tree

* A binary tree with a starting node called the root.
* Every tree is "rooted" if we know the starting/top node.

2. **Full Binary Tree**

* Every node has **0 or 2 children** — never just one.

### 3. **Complete Binary Tree**

* All levels are completely filled **except possibly the last**, and the last level nodes are **as far left as possible**.

### 4. **Perfect Binary Tree**

* A special kind of full tree where **all internal nodes have 2 children** AND **all leaves are at the same level**.

### 5. **Almost Complete Binary Tree**

* Another term often used **loosely** to describe a tree that is **"nearly complete"** — i.e., **complete with one or two nodes missing at the last level** (still left-aligned).

### 6. **Skewed Binary Tree**

* A tree where every node has only **one child**, making it look like a **linked list**.

Two types:

* **Left-skewed**: All nodes have only left children
* **Right-skewed**: All nodes have only right children

### 7. **Balanced Binary Tree**

* The height difference between the left and right subtree of any node is **at most 1**.

**Task 9:**

Applications of Graphs

**Answer:**

**Social Networks**

* Users are nodes; friendships/follows are edges.
* Example: Finding mutual friends, recommendations.

**Google Maps & GPS Navigation**

* Locations are nodes; roads are edges with weights (distance/time).
* Used for shortest path algorithms (like Dijkstra).

**Internet/Web Page Linking**

* Web pages as nodes; hyperlinks as directed edges.
* Used in PageRank and search engine indexing.

**Recommendation Systems**

* Users and products as nodes; purchases/likes as edges.
* Graph-based collaborative filtering.

**Network Routing**

* Routers as nodes; physical connections as edges.
* Graphs help in data routing, detecting failures.

**Dependency Resolution**

* Tasks/modules as nodes; dependencies as directed edges.
* Used in compilers, build systems (e.g., Maven, Gradle).

**AI Pathfinding (Games & Robotics)**

* Grid/cell-based maps are converted to graphs.
* Algorithms like A\* help find the shortest route.

**Task 10:**

Types of Graphs:

**Answer:**

**Directed Graph (Digraph)**

* Edges have a direction (from one node to another).
* Example: Twitter (A follows B, but B may not follow A).

**Undirected Graph**

* Edges have no direction; connection is mutual.
* Example: Facebook (A is friends with B and vice versa).

**Weighted Graph**

* Edges have weights (cost, distance, etc.).
* Example: Google Maps (edges represent roads with distance/time).

**Unweighted Graph**

* Edges have no weights.
* Example: Simple social networks or connections.

**Cyclic Graph**

* A path exists that starts and ends at the same node (a cycle).
* Example: Airline routes forming loops.

**Acyclic Graph**

* No cycles exist.
* Example: Task scheduling with dependencies.

**Connected Graph**

* There is a path between every pair of nodes (in undirected graph).

**Disconnected Graph**

* At least one node is unreachable from another.

**Complete Graph**

* Every node is directly connected to every other node.
* For n nodes, it has n(n-1)/2 edges (undirected).

**Sparse Graph**

* Has few edges compared to the number of nodes.

**Dense Graph**

* Has many edges — closer to the maximum possible.

**Tree (Special Graph)**

* A connected acyclic graph with n nodes and n-1 edges.
* Every tree is a graph, but not every graph is a tree.

**Task11:**

