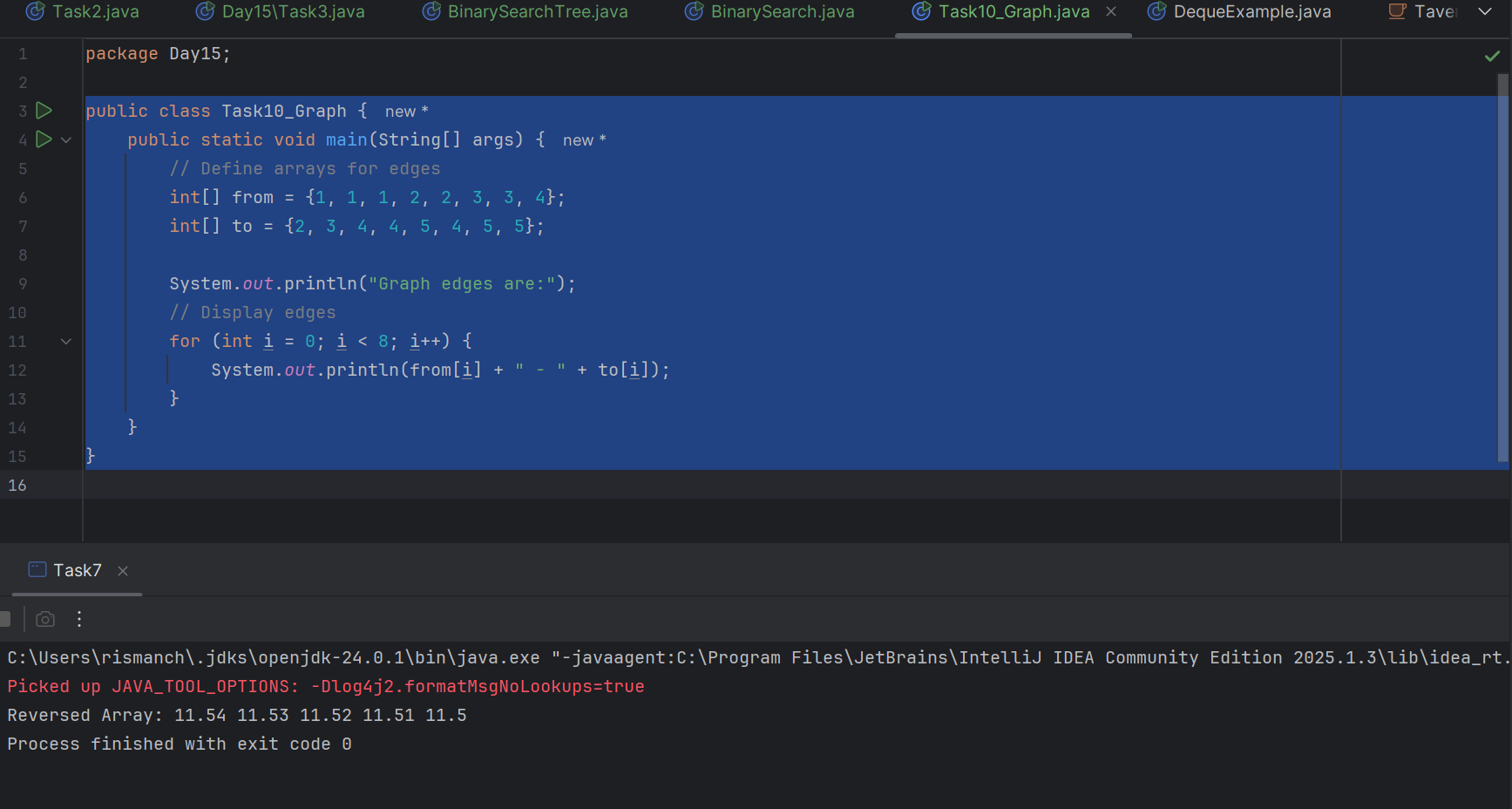
Name-Rishabh Manchanda

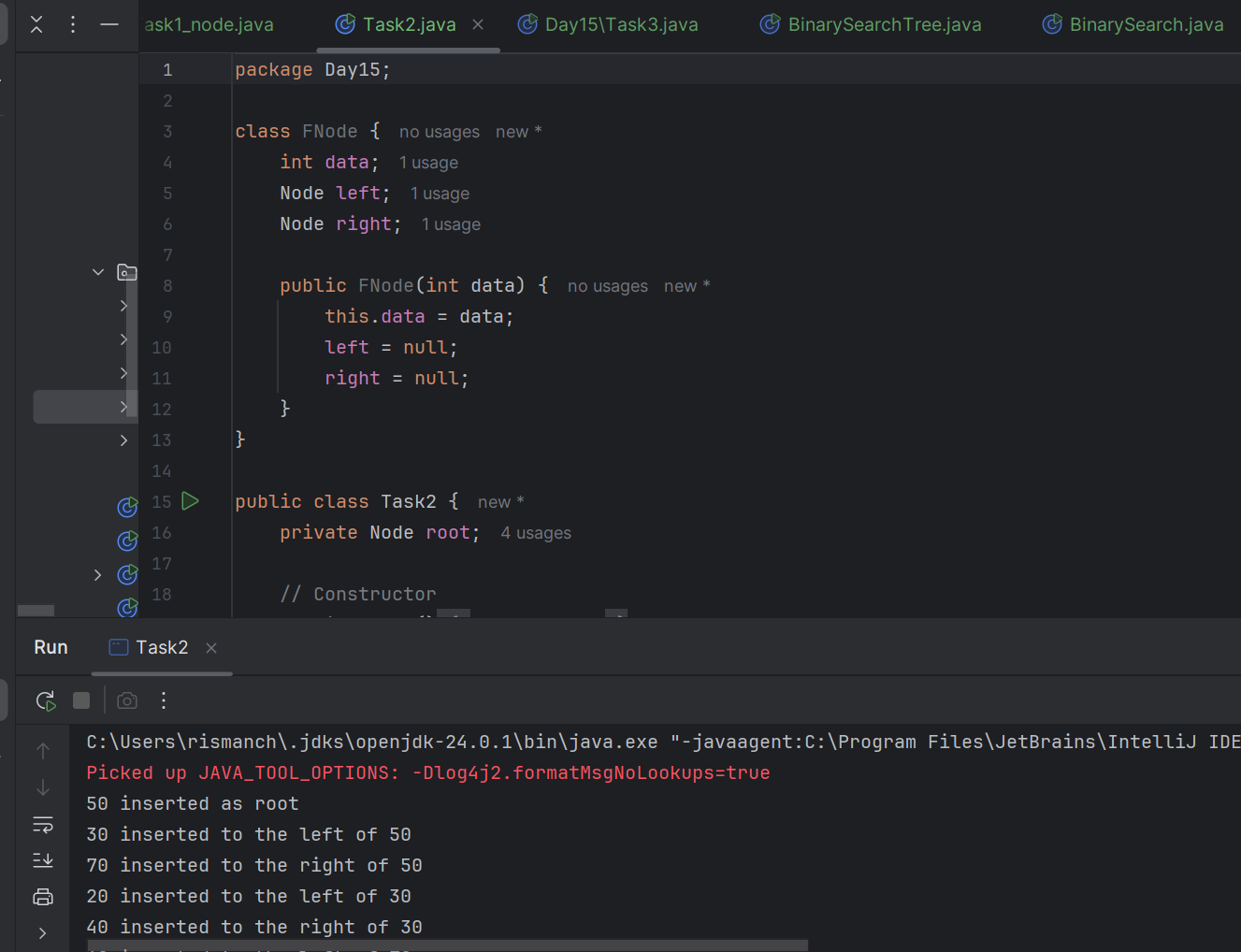
Employee id- Employee id- 109883998

Task -1  
  
Create a node for a tree and include a constructor.

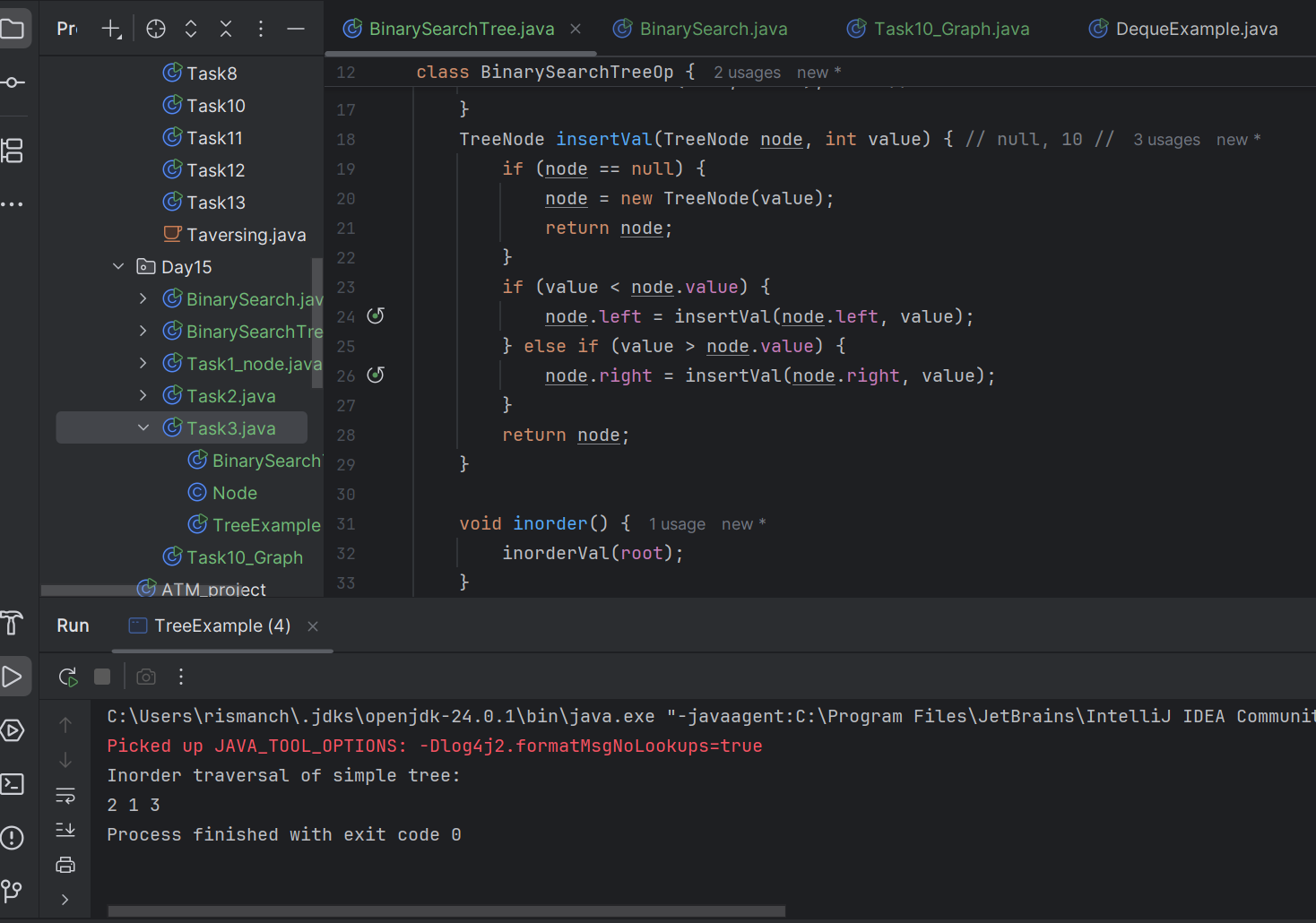


Task 2:

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



Task-3



Task 8:

Applications of Graphs

Here are the key applications of graphs across various domains:

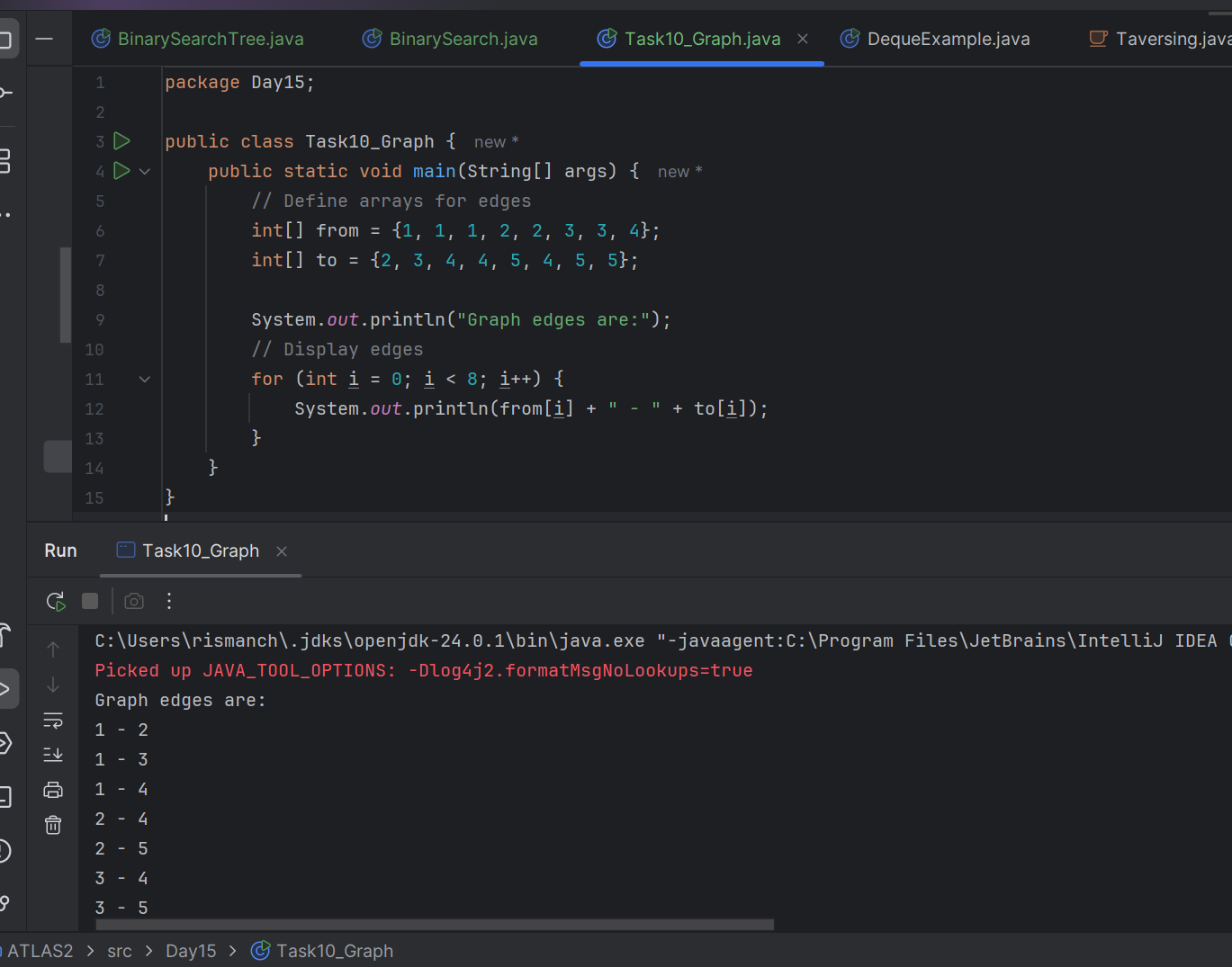
1. Social Networks
2. Computer Networks
3. Transportation and Navigation
4. Computer Science Applications
5. Telecommunications
6. Operations Research
7. Gaming and Entertainment

Task 9:

Types of Graphs:

1. Undirected Graph
2. Directed Graph
3. Weighted Graph
4. Complete Graph
5. Bipertite Graph
6. Tree
7. Cyclic Graph
8. Acyclic Graph
9. Regular Graph

Task-10



Q-What is the difference between recursion and iteration

Recursion and iteration are two fundamental programming concepts used to perform repetitive tasks. Here are the key differences between them:

### **1. Definition:**

* Recursion: A method in which a function calls itself to solve a problem. It typically involves a base case to terminate the recursive calls.
* Iteration: A process of repeating a set of instructions or statements using loops (like for, while, etc.) until a certain condition is met.

### **2. Structure:**

* Recursion: Involves function calls and can lead to multiple instances of the function being active on the call stack.
* Iteration: Involves looping constructs and maintains a single instance of the loop variable.

### **3. Memory Usage:**

* Recursion: Each recursive call consumes stack space, which can lead to a stack overflow if the recursion depth is too high.
* Iteration: Uses a fixed amount of memory for loop variables, making it generally more memory-efficient.

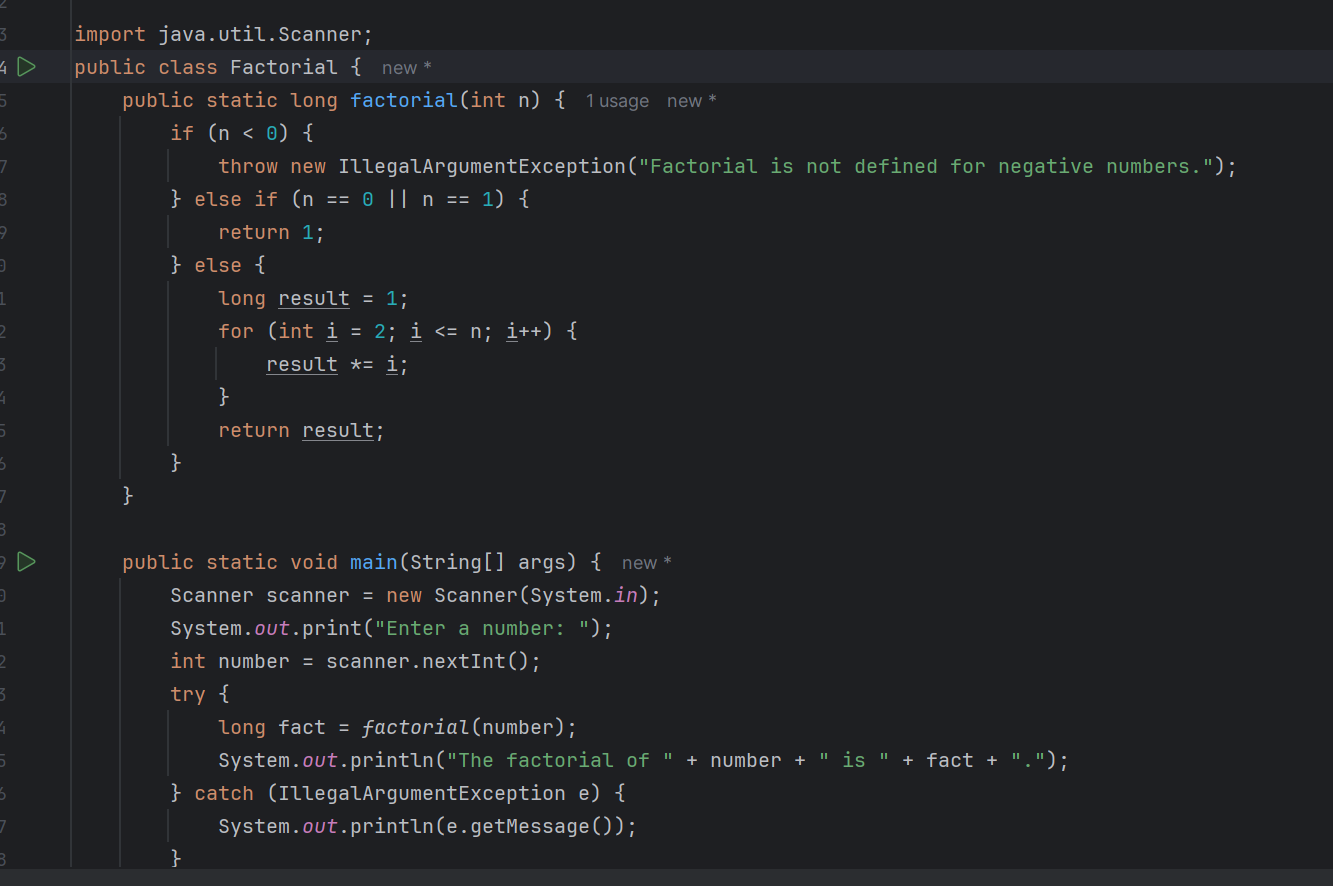
### **4. Performance:**

* Recursion: Can be less efficient due to the overhead of multiple function calls and context switching. However, it can lead to more elegant and simpler code for problems like tree traversals or factorial calculations.
* Iteration: Typically more efficient in terms of time and space, especially for simple repetitive tasks.

### **5. Use Cases:**

* Recursion: Often used in problems that can be divided into smaller subproblems, such as tree traversals, factorial calculation, and the Fibonacci series.
* Iteration: Commonly used for tasks that require repeated execution of a block of code, such as traversing arrays or lists.

Q-1 Wap to find the factorial of a number?



Wap to find the Fibonacci series of a number

