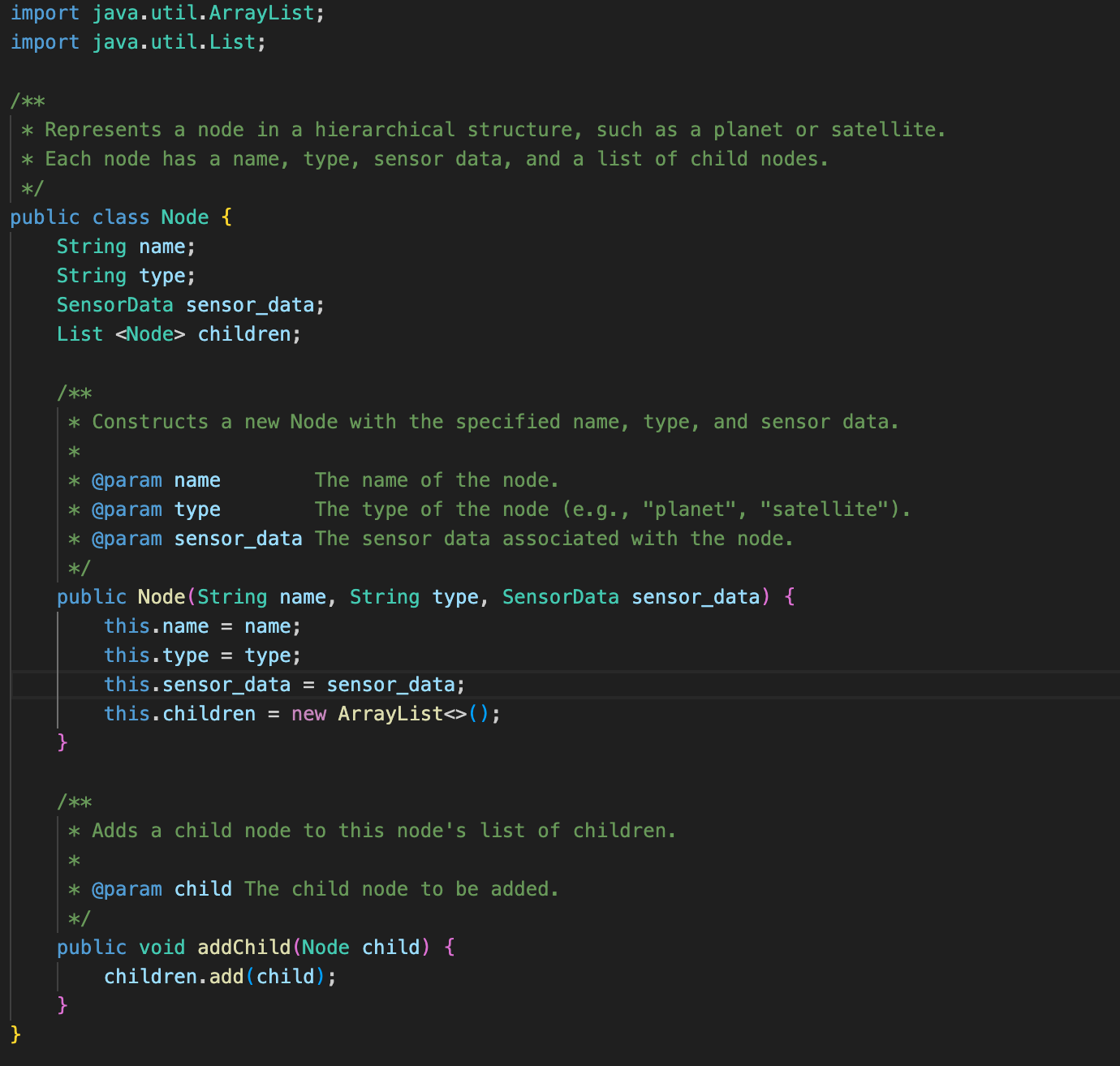
REPORT OF HOMEWORK 4 CSE22/505

Node Class:



The Node class represents objects like planets and satellites, and each node stores its name, type, and sensor data. Additionally, each node has a children list that holds the planets or satellites under it. The addChild method is used to add a child node to the current node. This class provides a fundamental structure to represent the hierarchical layout of the planetary system.

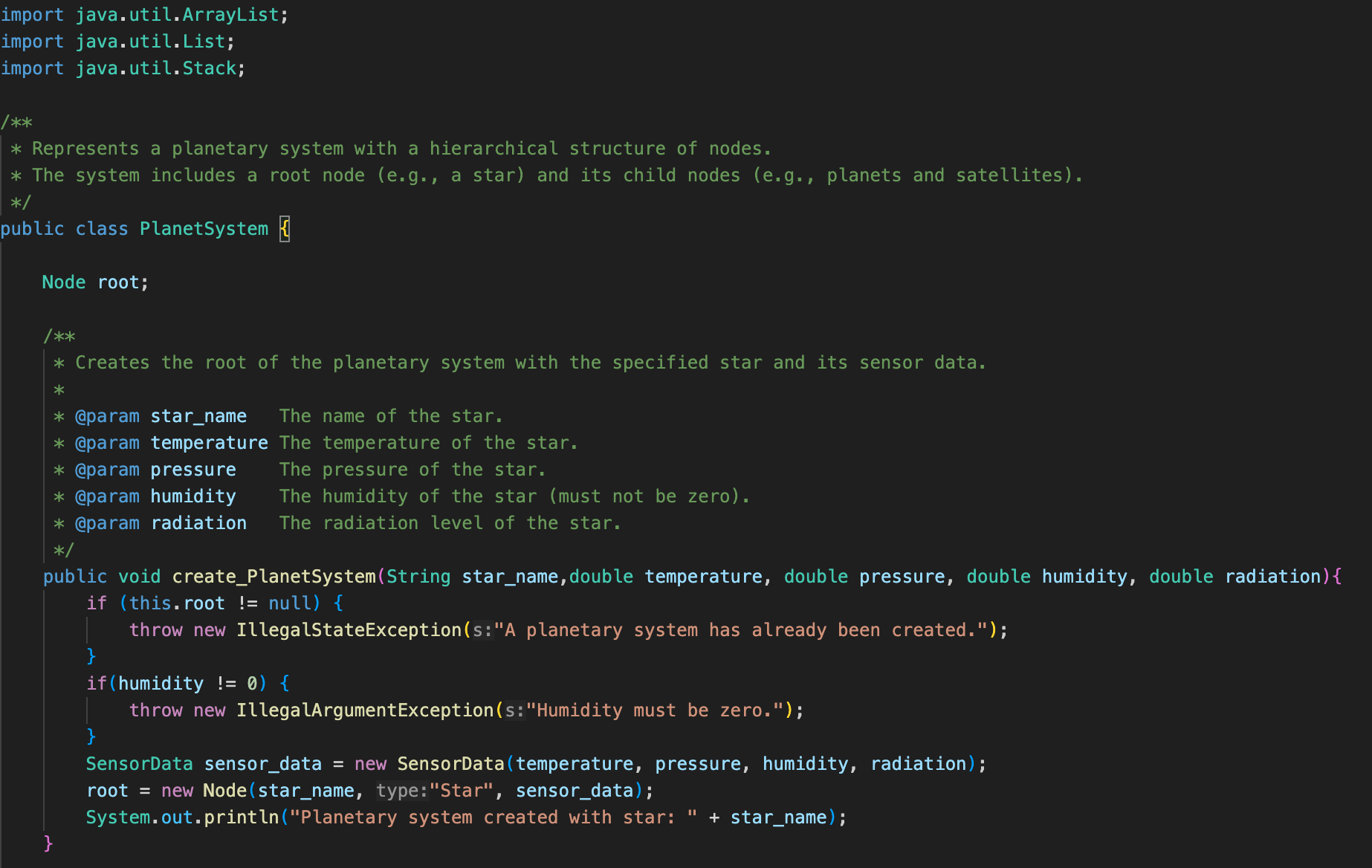
SensorData Class:



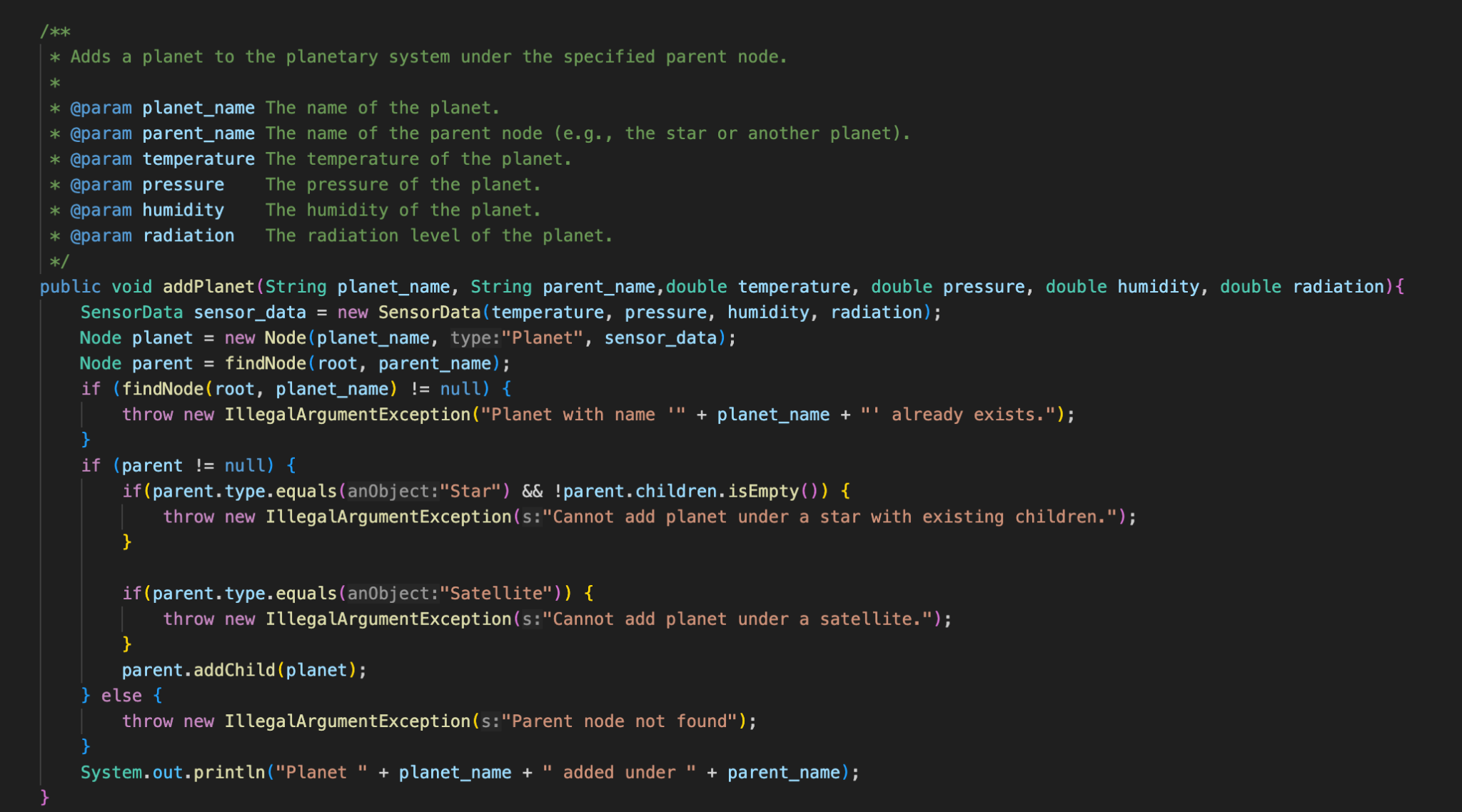
The SensorData class holds the sensor data for objects like planets and satellites. This data includes temperature, pressure, humidity, and radiation levels. The class checks if the parameters are valid, and if the humidity is not between 0 and 100, or if the temperature, pressure, or radiation is negative, it throws an exception with an appropriate error message. This class is used to organize and store the sensor data for each planet and satellite in a structured manner.

PlanetSystem Class:

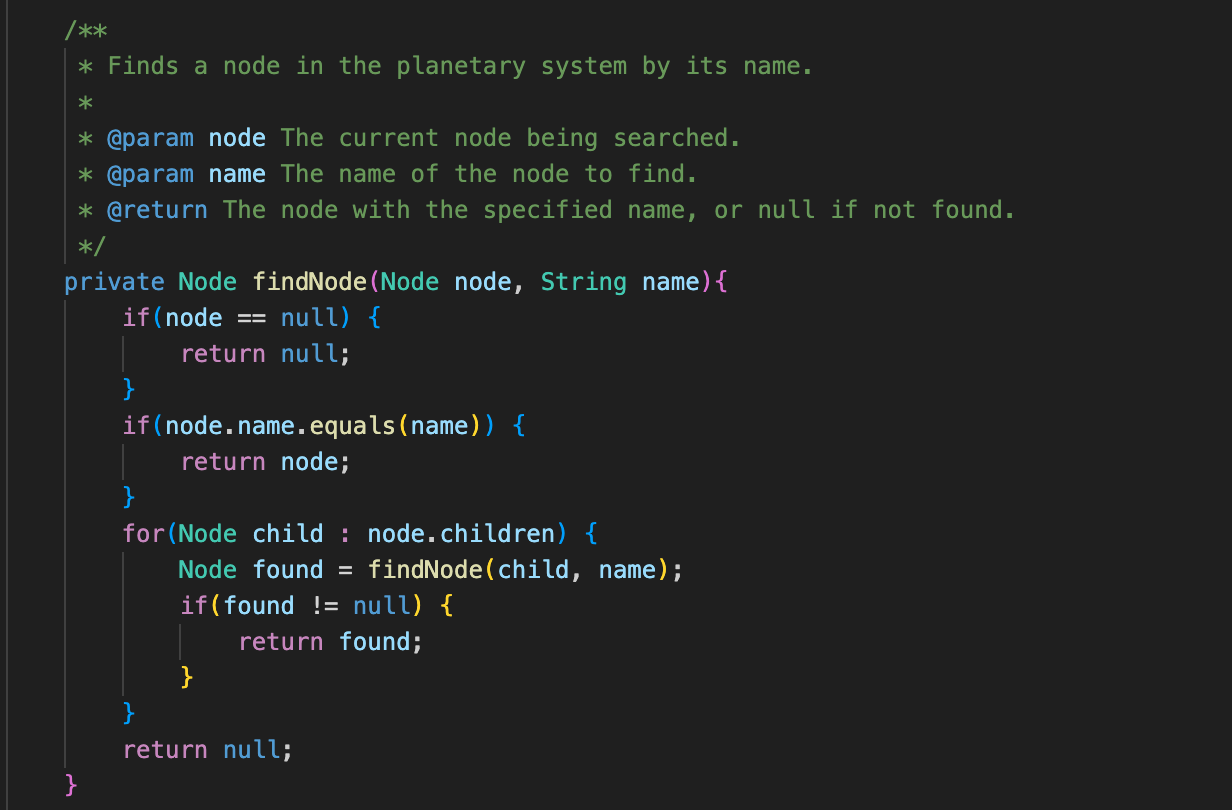
create\_PlanetSystem function:



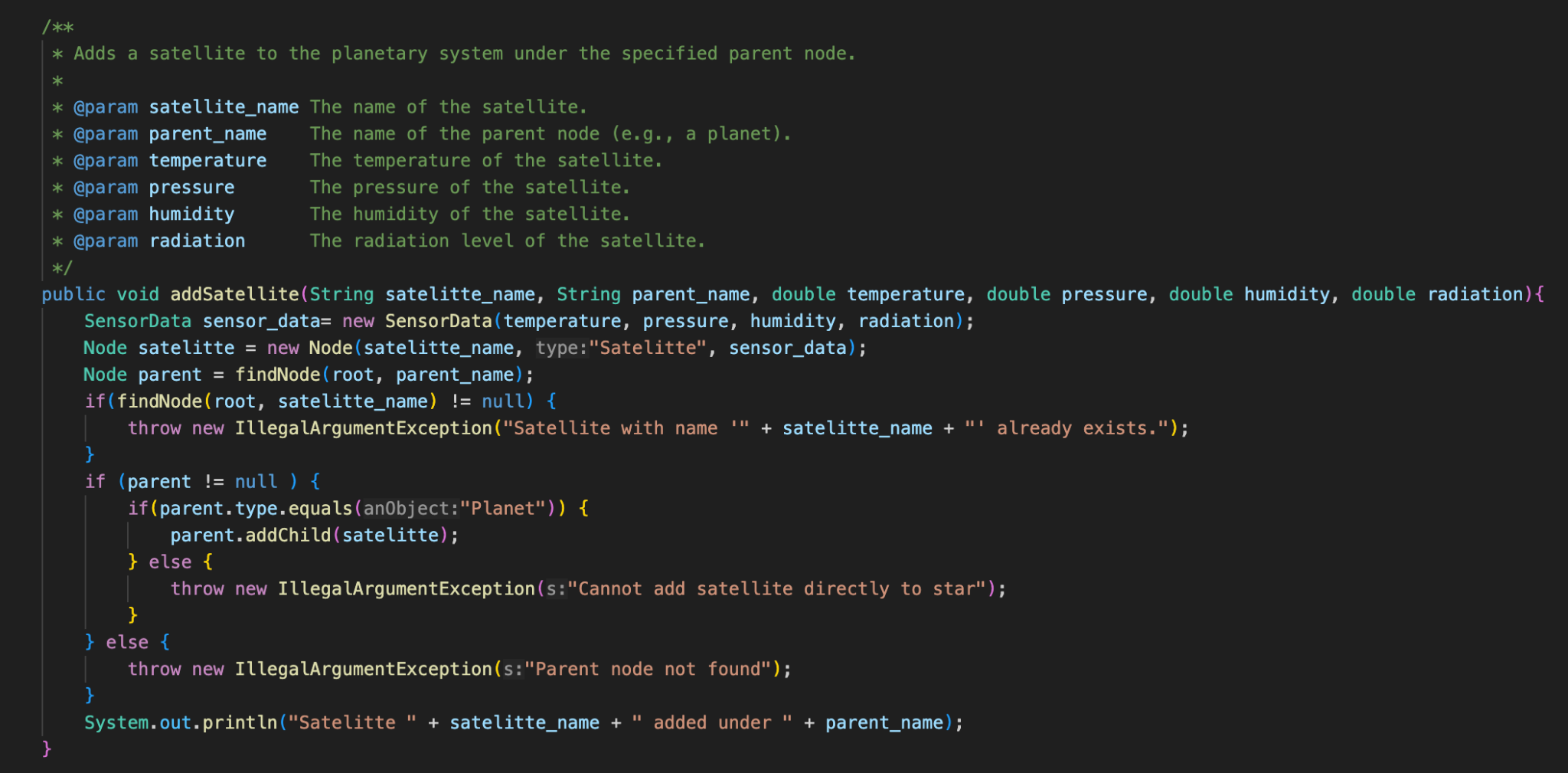
The **PlanetSystem** class represents the planetary system and manages the hierarchy between the root node (e.g., a star) and its child nodes. The **create\_PlanetSystem** function creates the root of the planetary system with the star's name, temperature, pressure, humidity, and radiation data. If a planetary system has already been created in the system, no new system can be created. Additionally, the humidity of the star must be zero; otherwise, an error is thrown. After creating the planetary system, this function sets the star as the root node.

addPlanet function:

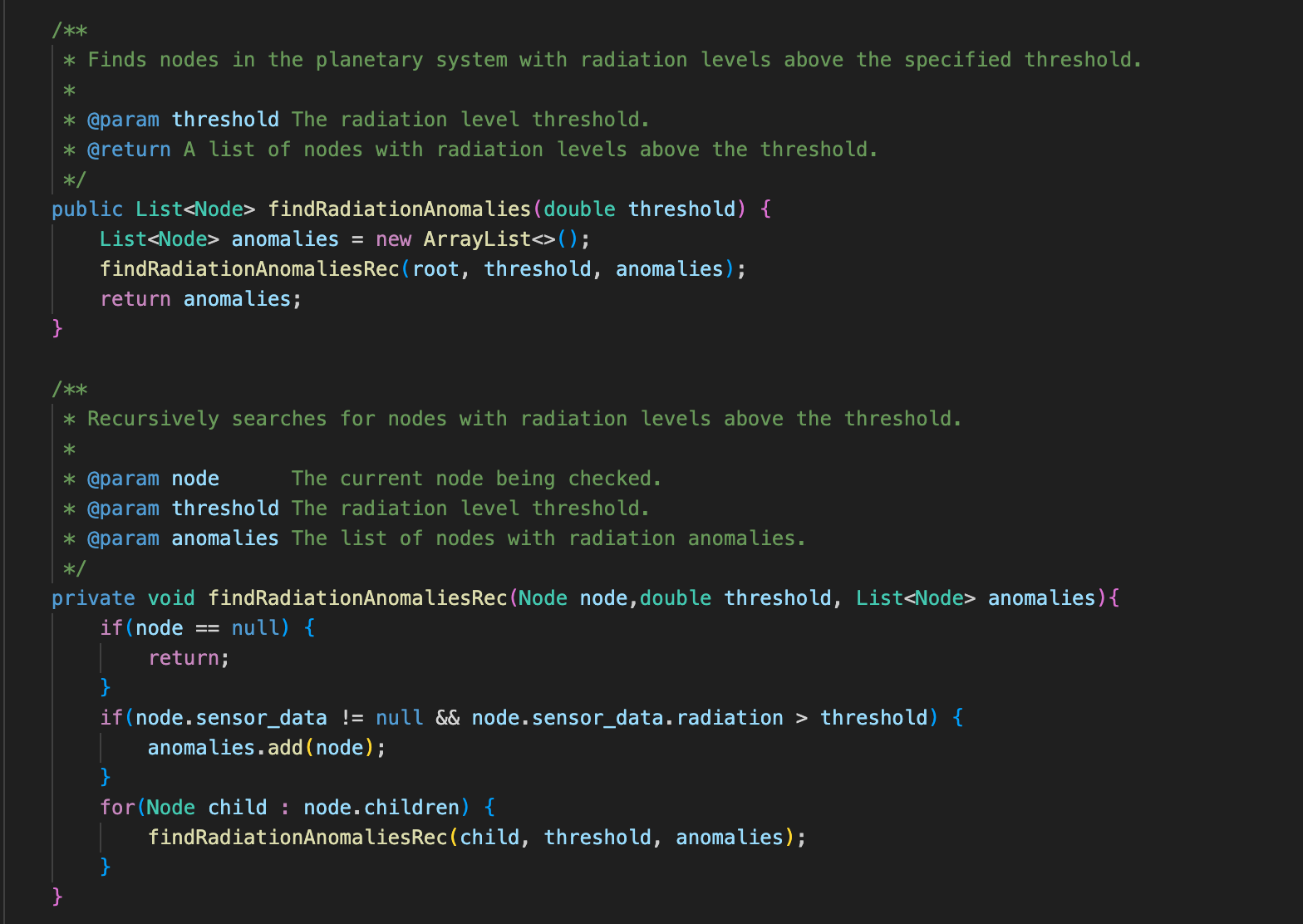
The addPlanet function adds a new planet under a specified parent node (either a star or a planet). First, a SensorData object containing the planet's sensor data is created, and then the planet's node is created. The findNode function checks if the specified parent node exists and verifies whether the planet has already been added to the system. If the planet has already been added or if the parent node is invalid, appropriate error messages are provided to the user. If all checks pass, the planet is added under the specified parent node. Additionally, when adding a planet under a star, if the star already has child planets or if an attempt is made to add the planet under a satellite, an error message is shown.

findNode function:

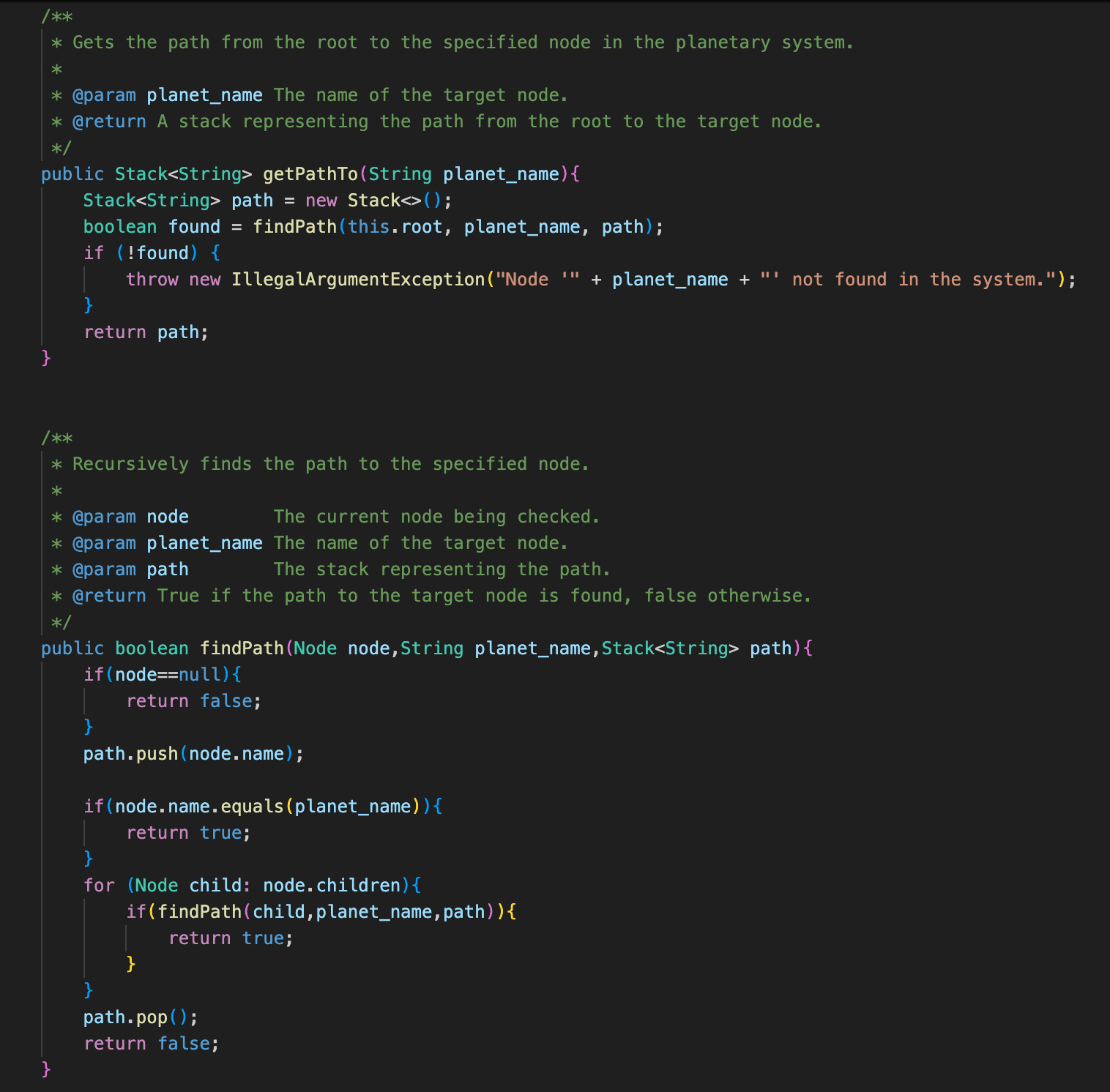
The findNode function searches for a node in the planetary system by its name. The function works recursively, starting from the root node and checking each child planet and satellite. If the node's name matches the search name, that node is returned. If the node is not found, the function checks all children, and if a match is found, that node is returned. As a result, if a node with the specified name is found, it is returned; otherwise, null is returned.

addSatellite function:

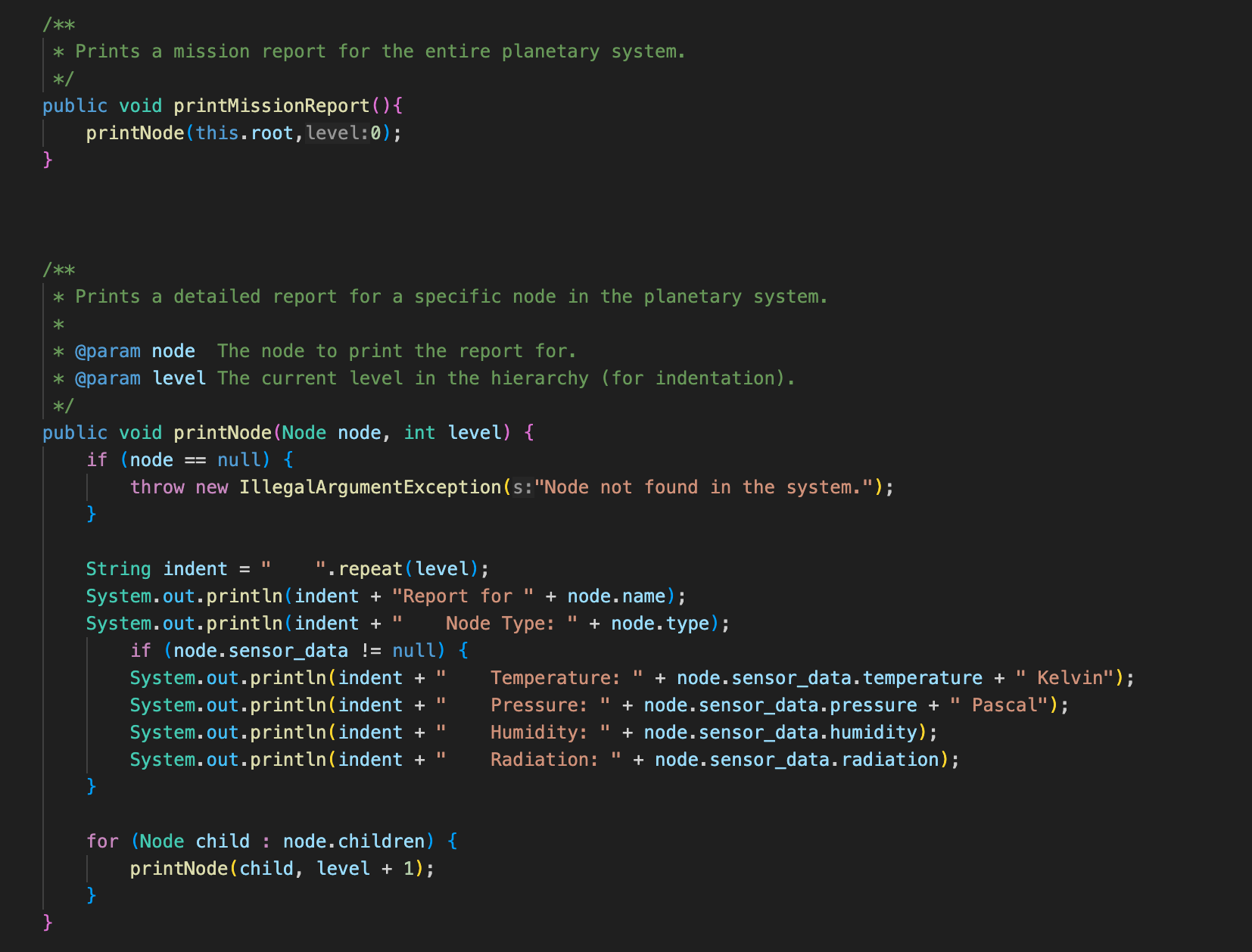
The addSatellite function adds a new satellite to a specified parent node. First, the sensor data for the satellite is created as a SensorData object. Then, a Node object is created with the satellite's name. The findNode function is used to find the specified parent node. If the satellite has already been added or if the parent node is of an incorrect type (satellites cannot be added under a star), an appropriate error message is shown. If all checks are successful, the satellite is added to the specified parent node.

findRadiationAnomalies and findRadiationAnomaliesRec function:

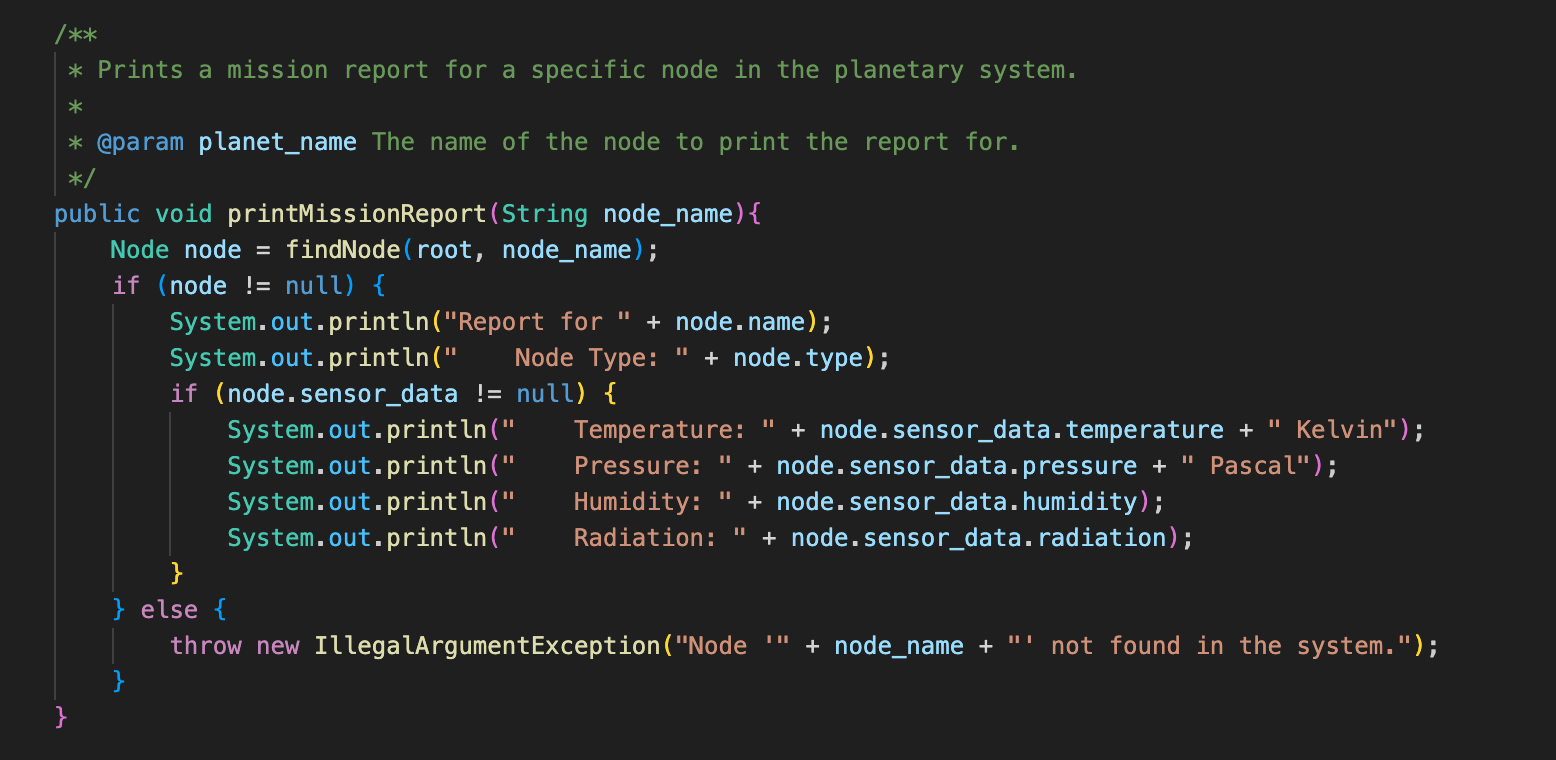
The findRadiationAnomalies function searches for nodes in the planetary system that exceed the specified radiation threshold. A recursive approach is used for this process. The function starts from the root node and checks the radiation level of each node. If a node's radiation level exceeds the specified threshold, that node is added to the anomalies list. The findRadiationAnomaliesRec helper function checks the radiation level for each node and recursively checks the child nodes in the same manner.

getPathTo and findPath function:

The getPathTo function is used to find the path to a specific planet or node. The planet\_name parameter provides the name of the target node in the system. The function calls the findPath function, which starts from the root node (the star) and finds the path to the target node, returning this path in a Stack data structure. If the target node is not found, the function throws an IllegalArgumentException.The findPath function works recursively, starting from the root node to find the path to the target node. This function checks the name of each node. If the node's name matches the target node's name, the path is added to the stack, and true is returned. If the target node is found in child nodes, the function recursively checks each child. If a child node reaches the target node, that child's name is added to the stack. If the target node is not found, the function returns false.

printMissionReport and printNode function:

The printMissionReport function is used to print the full report of the planetary system. This function calls the printNode function to print all the nodes in the planetary system. The printNode function uses the level parameter, which represents the level of each node, and calculates the appropriate indentation based on this level. This way, the hierarchy between planets and satellites is visually clarified.The name, type, and sensor data of each node are printed. If a node has child nodes, the function recursively repeats the same process for each child node. If the node is null, an error message stating "Node not found in the system" is thrown.

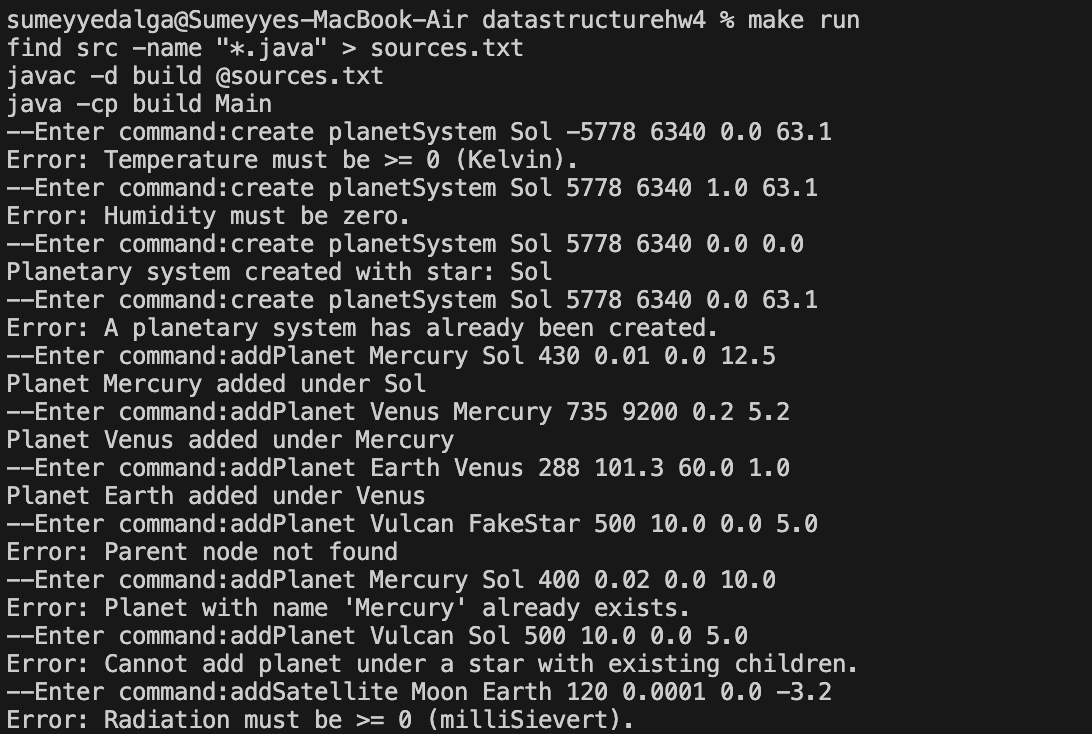
printMissionReport (with parameter) function:

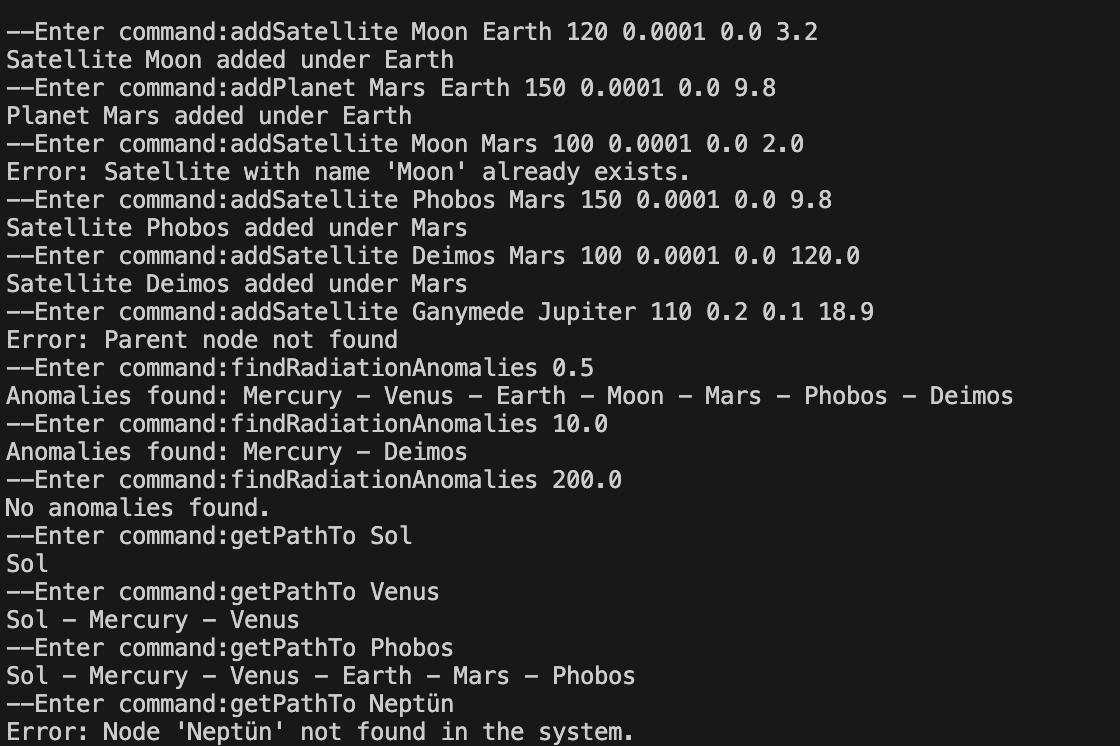
The printMissionReport function is used to print the report of a specific node. The function takes the planet\_name parameter, which provides the name of the target node, and searches for this node in the system using the findNode function. If the node is found, its name, type, and sensor data are printed to the screen. If the specified node cannot be found, an IllegalArgumentException is thrown, and an error message is displayed.

Main Class:

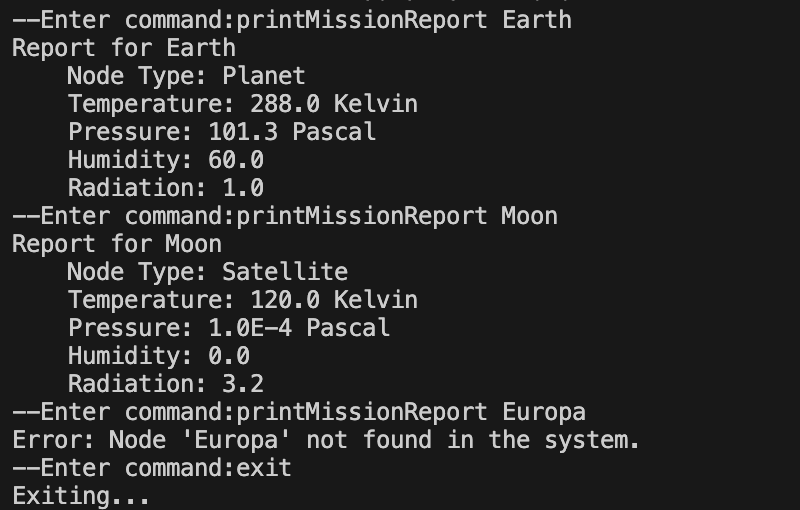
The Main class is the main class that runs the planetary system application. This class waits for the user to input commands and processes each command appropriately to perform operations on the planetary system. If an invalid command or parameter is entered, an appropriate error message is displayed. It also catches the exceptions thrown in the other classes.

OUTPUT:







****

**Sümeyye Dalga 220104004091**