You are tasked with building a simplified **File Directory System** using a Binary Search Tree (BST). Each node in the tree represents a file, identified by a unique file size (key) and a file name. Your implementation should support the following operations:

1. **Insert a file**: Add a new file to the directory based on its file size.
2. **Delete a file**: Remove a file from the directory given its file size.
3. **Find the smallest file**: Return the name and size of the smallest file in the directory.
4. **Find the largest file**: Return the name and size of the largest file in the directory.
5. **In-order traversal**: List all files in ascending order of their sizes.

Code Template

```
class FileNode {
    int fileSize; // Size of the file (key)
    String fileName; // Name of the file
    FileNode left, right;

    public FileNode(int fileSize, String fileName) {
        this.fileSize = fileSize;
        this.fileName = fileName;
        this.left = null;
        this.right = null;
    }
}
class FileDirectory {
    private FileNode root;

    public FileDirectory() {
        this.root = null;
    }

    // TODO: Implement the insert method
    public void insert(int fileSize, String fileName) {
        // Add code here
    }

    // TODO: Implement the delete method
    public void delete(int fileSize) {
        // Add code here
    }

    // TODO: Implement the method to find the smallest file
    public FileNode findMin() {
        // Add code here
        return null; // Placeholder
    }

    // TODO: Implement the method to find the largest file
    public FileNode findMax() {
        // Add code here
        return null; // Placeholder
    }

    // TODO: Implement in-order traversal
    public void inOrderTraversal(FileNode node) {
        // Add code here
    }

    // Helper method to start traversal
    public void displayFiles() {
        inOrderTraversal(this.root);
    }
}
```

```java
public class FileDirectorySystem {
    public static void main(String[] args) {
        FileDirectory directory = new FileDirectory();

        // Sample files to be inserted
        directory.insert(150, "FileA");
        directory.insert(300, "FileB");
        directory.insert(100, "FileC");
        directory.insert(200, "FileD");
        directory.insert(50, "FileE");

        // Test in-order traversal
        System.out.println("Files in ascending order of size:");
        directory.displayFiles();

        // Find the smallest file
        FileNode smallest = directory.findMin();
        System.out.println("Smallest file: " + smallest.fileName + " (" + smallest.fileSize + " KB)");

        // Find the largest file
        FileNode largest = directory.findMax();
        System.out.println("Largest file: " + largest.fileName + " (" + largest.fileSize + " KB)");

        // Test delete operation
        System.out.println("Deleting file with size 100 KB.");
        directory.delete(100);
        System.out.println("Files after deletion:");
        directory.displayFiles();
    }
}
```

Example output

```
Files in ascending order of size:
FileE (50 KB)
FileC (100 KB)
FileA (150 KB)
FileD (200 KB)
FileB (300 KB)

Smallest file: FileE (50 KB)
Largest file: FileB (300 KB)

Deleting file with size 100 KB.
Files after deletion:
FileE (50 KB)
FileA (150 KB)
FileD (200 KB)
FileB (300 KB)
```