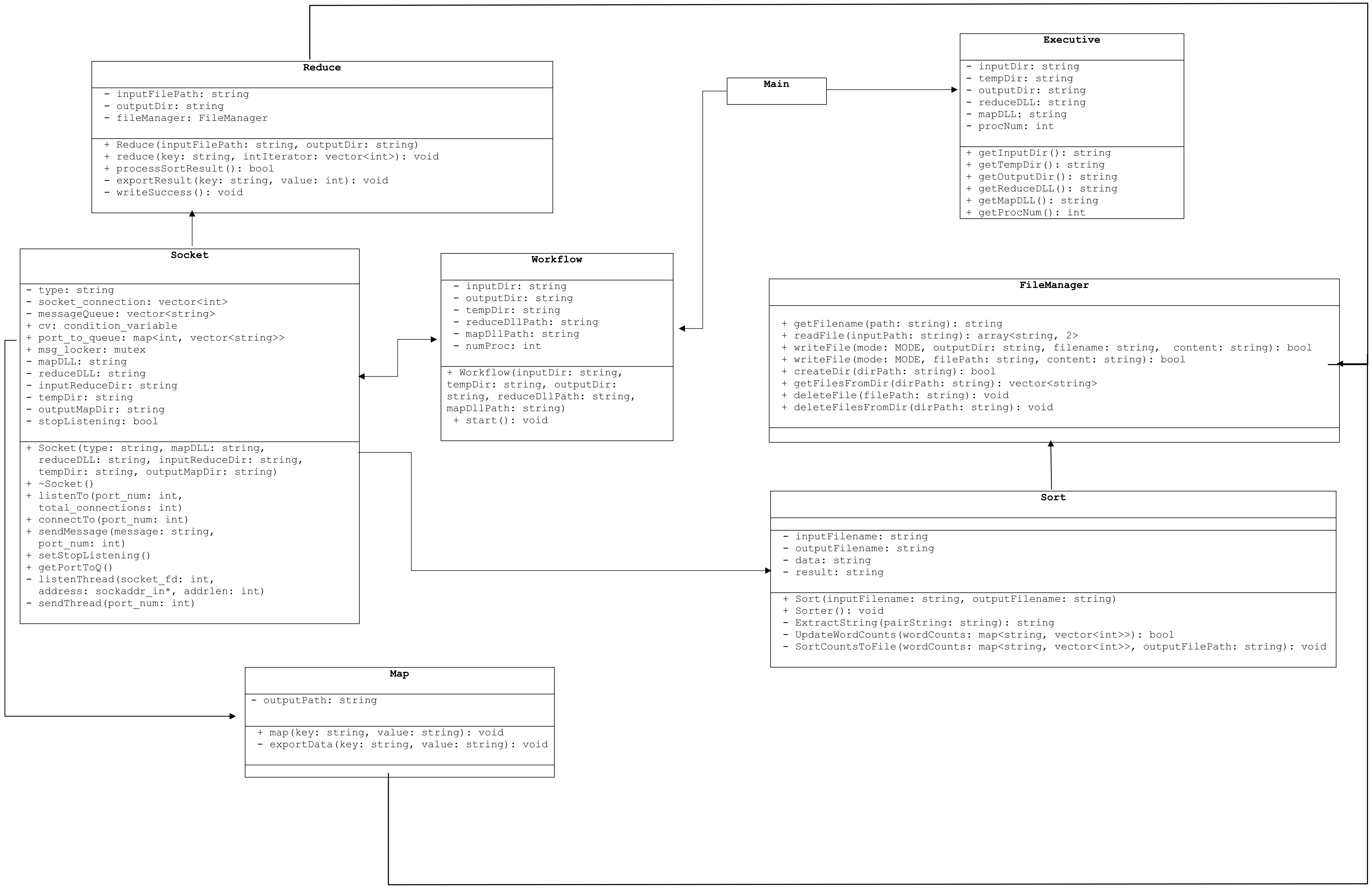
Design Document

Karim Elbourakkadi
David Saie
Matt Cockle
Varis Nijat

1. The Executive class is called first, as it serves as the entry point for the program. It takes command-line arguments as inputs, including the input directory, temporary directory, output directory, reduce DLL path, and map DLL path. These arguments are used to initialize its member variables. Inputs: Command-line arguments specifying the number of processes, input directory, temporary directory, output directory, reduce DLL path, and map DLL path. Outputs: None directly, but it initializes and sets the necessary parameters for the Workflow class.
2. The main class interacts with the Executive and Workflow class. It instantiates the Workflow class with the provided input directory, temporary directory, output directory, reduce DLL path, and map DLL path provided by the Executive class. Then, it calls the start() method of the Workflow class. Inputs: Number of processes, Input directory, temporary directory, output directory, reduce DLL path, and map DLL path. Outputs: None directly, but it triggers the execution of the workflow.
3. The Workflow class triggers the execution of the Socket class, which facilitates socket communication. It allows the creation of sockets, listening for connections, connecting to existing sockets, and sending/receiving messages. The class utilizes threads to handle concurrent listening and sending of messages.
4. The Socket class interacts with other entities such as the Controller, Stubs, and Threads (Map & Reduce). The Controller and Stubs call the listenTo method of the Socket class to create sockets and listen for connections on specified ports. The listenTo method utilizes multiple threads to handle concurrent connections, allowing the Socket class to listen for incoming messages from multiple sources simultaneously. The Threads (Map & Reduce) use the connectTo method of the Socket class to connect to pre-existing sockets. The connectTo

method also utilizes threads to establish connections concurrently, enabling efficient communication between the Threads and the Socket class. Objects that use the connectTo method can then send messages using the sendMessage method to the established connection.

5. The Socket class interacts with the Map class. It instantiates the Map class with the output path and calls the map() function of the Map class. The map() function tokenizes the input data into distinct words and exports the tokenized values to temporary files. Inputs: Output path, input data (key-value pairs). Outputs: Tokenized values exported to temporary files.
6. Next, the Socket class interacts with the Sort class. It instantiates the Sort class with the input filename and output filename. Then, it calls the Sorter() function of the Sort class. The Sorter() function reads the input file, updates word counts, and sorts the word counts. The sorted output is written to the output file. Inputs: Input filename, output filename. Outputs: Sorted word counts written to the output file.
7. The socket class interacts with the Reduce class. It instantiates the Reduce class with the input file path and output directory. Then, it calls the reduce() function of the Reduce class. The reduce() function processes the sorted result and exports the reduced data to a text file. Inputs: Input file path, output directory. Outputs: Reduced data exported to a text file.


Reduce
- inputFilePath: string
- outputDir: string
- fileManager: FileManager
+ Reduce(inputFilePath: string, outputDir: string)
+ reduce(key: string, intIterator: vector<int>): void
+ processSortResult(): bool
- exportResult(key: string, value: int): void
- writeSuccess(): void
Main
Executive
- inputDir: string
- tempDir: string
- outputDir: string
- reduceDLL: string
- mapDLL: string
- procNum: int
+ getInputDir(): string
+ getTempDir(): string
+ getOutputDir(): string
+ getReduceDLL(): string
+ getMapDLL(): string
+ getProcNum(): int
Socket
- type: string
- socket_connection: vector<int>
- messageQueue: vector<string>
+ cv: condition_variable
+ port_to_queue: map<int, vector<string>>
+ msg_locker: mutex
- mapDLL: string
- reduceDLL: string
- inputReduceDir: string
- tempDir: string
- outputMapDir: string
- stopListening: bool
+ Socket(type: string, mapDLL: string, reduceDLL: string, inputReduceDir: string, tempDir: string, outputMapDir: string)
+ ~Socket()
+ listenTo(port_num: int, total_connections: int)
+ connectTo(port_num: int)
+ sendMessage(message: string, port_num: int)
+ setStopListening()
+ getPortToQ()
- listenThread(socket_fd: int, address: sockaddr_in*, addrlen: int)
- sendThread(port_num: int)
Workflow
- inputDir: string
- outputDir: string
- tempDir: string
- reduceDllPath: string
- mapDllPath: string
- numProc: int
+ Workflow(inputDir: string, tempDir: string, outputDir: string, reduceDllPath: string, mapDllPath: string)
+ start(): void
FileManager
+ getFilename(path: string): string
+ readFile(inputPath: string): array<string, 2>
+ writeFile(mode: MODE, outputDir: string, filename: string, content: string): bool
+ writeFile(mode: MODE, filePath: string, content: string): bool
+ createDir(dirPath: string): bool
+ getFilesFromDir(dirPath: string): vector<string>
+ deleteFile(filePath: string): void
+ deleteFilesFromDir(dirPath: string): void
Sort
- inputFilename: string
- outputFilename: string
- data: string
- result: string
+ Sort(inputFilename: string, outputFilename: string)
+ Sorter(): void
- ExtractString(pairString: string): string
- UpdateWordCounts(wordCounts: map<string, vector<int>>): bool
- SortCountsToFile(wordCounts: map<string, vector<int>>, outputFilePath: string): void
Map
- outputPath: string
+ map(key: string, value: string): void
- exportData(key: string, value: string): void