**Host 1 (Ezkylle):**

**"Hello Everyone Hello Sir I’m Ezkylle Opiniano, Today we break down our code into bite-sized pieces. I’m Ezkylle, and today we’ve got something interesting for you: a Java-based Delivery Agent Management System."**

**Host 2 (Adrian):**

**"Hey everyone, I’m Adrian San Pedro! This system handles everything from managing delivery agents, their vehicles, the types of goods they transport, and saving all that data to a CSV file."**

**Host 3 (Marco):**

**"Yup! I’m Marco Manio, and we’ll be diving into this system one part at a time, explaining how each class and method works. So, let’s get into it!"**

**Part 1: The DeliveryAgent Class**

**Ezkylle (Host 1):**

**"Alright, I’ll kick things off by talking about the core of this system the DeliveryAgent class. This is the class that represents each delivery agent in the system. It has attributes like agentID, vehicleType, goodsType, and availability. But here’s a neat part: it also extends from a class called Person."**

**Adrian (Host 2):**

**"Right, the Person class is pretty straightforward. It just defines the basic properties every person has, like name and contact number. So, when DeliveryAgent extends Person, it automatically inherits those properties."**

**Ezkylle (Host 1):**

**"Exactly. So, if you take a look at the constructor in DeliveryAgent, it looks like this:**

**(SHOW CODE FROM THE TEAMS)**

**The super() call passes the name and contact number to the Person class, and the rest of the constructor sets the delivery agent-specific details like their vehicle and availability."**

**Marco (Host 3):**

**"So basically, it’s like saying, ‘Hey, I’ve got all this data for a new delivery agent,’ and the constructor sets everything up for you. And then, of course, there are getter and setter methods to access and update those values."**

**Adrian (Host 2):**

**"Yep, and that’s important. If you ever want to update an agent’s vehicle or check if they’re available, you use methods like getVehicleType() or isAvailability()."**

**Part 2: CSV File Handling**

**Marco (Host 3):**

**"Alright, my turn. Let’s talk about how all of this data gets stored and retrieved. That’s where the CSVUtils class comes in. In this system, we save agent data to a CSV file and then load it back when needed."**

**Adrian (Host 2):**

**"Yeah, CSVs are super common for storing simple data. What does the CSVUtils class do with them?"**

**Marco (Host 3):**

**"Good question! It has two main methods: writeToCSV() and readFromCSV(). So, writeToCSV() takes a list of agents, converts their details into a comma-separated format, and writes it to the file. Here’s a sample of what the output might look like in the CSV:**

**1, John Doe,1234567890, Food, Car, true**

**That’s the agent’s ID, name, contact number, the type of goods they carry, the vehicle they use, and whether they’re available. All separated by commas."**

**Ezkylle (Host 1):**

**"Nice! And then when we want to load those agents back, we use readFromCSV(), right?"**

**Marco (Host 3):**

**"Exactly. readFromCSV() reads each line from the CSV file, splits it by commas, and uses the fromCSV() method in DeliveryAgent to recreate each agent object. It converts the plain text back into Java objects so the system can work with them."**

**Part 3: DeliveryAgentManager**

**Adrian (Host 2):**

**"Alright, time for the DeliveryAgentManager class! This class is the ‘boss’ of the system. It handles the list of agents and all the operations you can perform on them, like adding, updating, or removing an agent."**

**Ezkylle (Host 1):**

**"So, it’s like the manager that keeps everything in check?"**

**Adrian (Host 2):**

**"Exactly! The DeliveryAgentManager interacts with CSVUtils to load agents from the file and save changes. If you want to add a new agent, call addAgent(). The manager takes care of storing that new agent in the list and saving them to the CSV file."**

**Marco (Host 3):**

**"And it can also remove agents, right? If you want to delete someone, you pass in the agent’s ID, and it takes care of it with removeAgent()."**

**Adrian (Host 2):**

**"Yep, and if you want to update an agent’s details — like their vehicle type or availability — you can call updateAgent(). The manager finds the agent by their ID and updates their information."**

**Part 4: Filtering and Searching Agents**

**Ezkylle (Host 1):  
"Another cool feature in this system is the ability to filter and search agents. The DeliveryAgentManager has methods for filtering agents by their vehicle type and searching by the type of goods they carry."**

**Marco (Host 3):  
"So, let’s say you only want to see agents that drive a truck. You would use filterByVehicleType(), and it would return a list of agents who use a truck."**

**Adrian (Host 2):  
"And the search feature works the same way. You can search for agents by the type of goods they transport using searchByGoodsType(). It returns agents carrying that specific kind of goods."**

**Part 5: The Main Program and User Interface**

**Marco (Host 3):**

**"Finally, let’s talk about how it all comes together in the Main class. This is where the program runs. It creates an instance of DeliveryAgentManager and kicks off a loop where the user can interact with the system."**

**Adrian (Host 2):**

**"Right! The user interface is pretty simple it displays a menu of options like adding an agent, updating them, or listing all the agents. The user selects an option, and the corresponding method from DeliveryAgentManager is called to act."**

**Ezkylle (Host 1):**

**"And after each action, like adding or updating an agent, the system automatically saves the data to the CSV file. That way, you don’t lose any chances."**

**Closing Remarks**

**Marco (Host 3):**

**"So, to sum it up, this system does a lot! We’ve got the DeliveryAgent class for individual agents, CSVUtils for file handling, DeliveryAgentManager for managing the agents, and the Main class for running the program and handling user interaction."**

**Adrian (Host 2):**

**"Yeah, it’s a full-fledged system for managing delivery agents, and it shows how you can use object-oriented programming and file handling to build something practical."**

**Ezkylle (Host 1):**

**"That’s right! We hope you found this breakdown useful. Whether you’re learning Java or working on your projects, this system gives you a solid foundation to build on."**

**Adrian (Host 2):**

**"Absolutely. Thanks for listening!”**

**Marco (Host 3):**

**"See you next time!"**