Dividing the Address Manager Code into Tasks:

Here's a breakdown of the code for the Address Manager application into smaller, manageable tasks:

Model Tasks:

1. **Task 1.1:** Define the `Address.java` class with member variables for name, street, city, postal code, etc.
2. **Task 1.2:** Implement getters and setters for each member variable in `Address.java`. (Optional)
3. **Task 1.3:** Create a constructor in `Address.java` to initialize the address object (Optional).

Controller Tasks:

1. **Task 2.1:** Define a method in `AddressController.java` to add a new address:
    - Take user input for address details.
    - Create a new `Address` object using the input data.
2. **Task 2.2:** Define a method in `AddressController.java` to edit an existing address:
    - Allow user selection of the address to edit.
    - Take user input for modifications.
    - Update the corresponding `Address` object.
3. **Task 2.3:** Define a method in `AddressController.java` to delete an address:
    - Allow user selection of the address to delete.
    - Remove the chosen address object from storage.
4. **Task 2.4:** Define a method in `AddressController.java` to retrieve all addresses:
    - Fetch all addresses from storage (in-memory list, file, or database).
    - Prepare the retrieved addresses for display in the view.
5. **Task 2.5 (Optional):** Implement methods in `AddressController.java` for JSON communication:
    - Serialize an `Address` object to a JSON string.
    - Deserialize a JSON string back to an `Address` object.

Data Storage Tasks (Choose one):

1. **Task 3.1 (Simple):** Define methods in `AddressController.java` to manage addresses in an in-memory list (using ArrayList).

2. **Task 3.2 (Intermediate):** Implement methods in `AddressController.java` to:
    - Read addresses from a JSON file on disk during application startup.
    - Write updated addresses to a JSON file upon changes.
3. **Task 3.3 (Advanced):** Establish connection to a database (MySQL, MongoDB) in `AddressController.java`.
    - Implement methods to store, retrieve, and update addresses in the database.

View Tasks (Depending on UI framework):

1. **Task 4.1:** Design the user interface layout for displaying a list of addresses.
2. **Task 4.2:** Create UI components for adding a new address (input fields for details).
3. **Task 4.3:** Develop UI elements for editing existing addresses (allow modifications).
4. **Task 4.4:** Implement functionality to delete chosen addresses from the view.
5. **Task 4.5:** Connect UI elements (buttons, text fields) to the corresponding controller methods using action listeners or event handlers.

Additional Tasks:

1. **Task 5.1:** Implement error handling in controller methods for invalid user input or data storage issues. (Optional)
2. **Task 5.2:** Design search functionality in the view to find specific addresses (by name, city, etc.). (Optional)
3. **Task 5.3:** Implement data validation on the view side to ensure users enter data in the correct format. (Optional)