Courier Management System

Coding Task 8

Collections

Scope: ArrayList/Hashmap

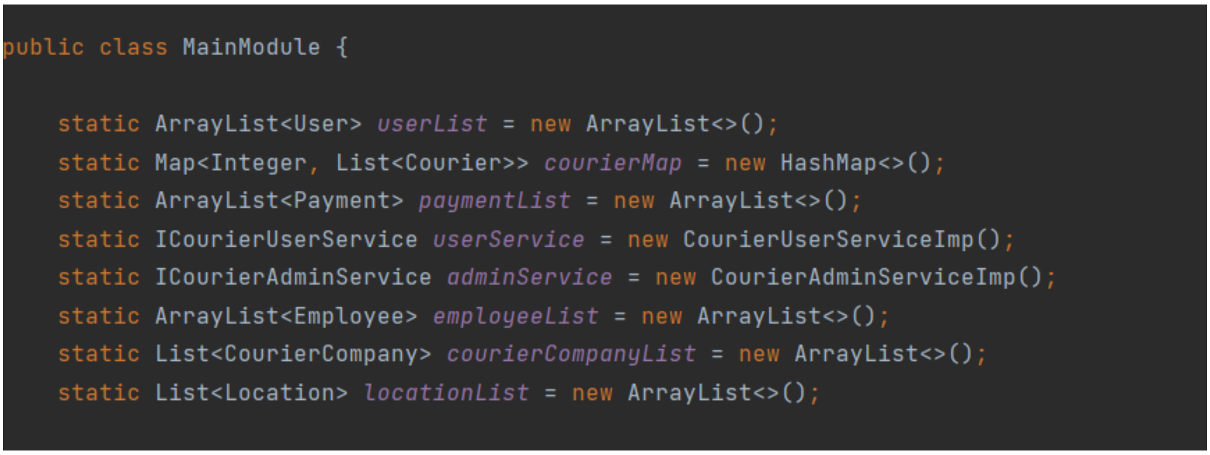
Task: Improve the Courier Management System by using Java collections:

1. Create a new model named CourierCompanyCollection in entity package replacing the Array of Objects with List to accommodate dynamic updates in the CourierCompany class

2. Create a new implementation class CourierUserServiceCollectionImpl class in package dao which implements ICourierUserService interface which holds a variable named companyObj of type CourierCompanyCollection

This task aims to enhance the **Courier Management System** by incorporating Java Collections such as ArrayList and HashMap to enable dynamic data handling. Instead of using fixed-size arrays, collections provide better flexibility and scalability for real-time data operations like insertions, deletions, and lookups.

In my Courier Management System, I have used **Java Collections**, specifically ArrayList and HashMap, to store and manage all the data dynamically. Instead of relying on static arrays, I used ArrayList to handle lists such as users, employees, courier companies, payments, and locations. For mapping courier orders to users, I used a HashMap with the user ID as the key and a list of their couriers as the value. This approach allowed me to add, update, and retrieve data efficiently, making the system more flexible, scalable, and easier to maintain.



```
public class MainModule {

    static ArrayList<User> userList = new ArrayList<>();
    static Map<Integer, List<Courier>> courierMap = new HashMap<>();
    static ArrayList<Payment> paymentList = new ArrayList<>();
    static ICourierUserService userService = new CourierUserServiceImp();
    static ICourierAdminService adminService = new CourierAdminServiceImp();
    static ArrayList<Employee> employeeList = new ArrayList<>();
    static List<CourierCompany> courierCompanyList = new ArrayList<>();
    static List<Location> locationList = new ArrayList<>();
```

Collection Usage in MainModule

In the MainModule class, various collections were initialized to handle real-time data:

- ArrayList<User> userList – stores all registered users.

- HashMap<Integer, List<Courier>> courierMap – maps user IDs to their corresponding courier orders.
- ArrayList<Payment> paymentList – keeps track of payment records.
- ArrayList<Employee> employeeList – holds all employee details.
- List<CourierCompany> courierCompanyList – manages available courier companies.
- List<Location> locationList – contains delivery locations.

These collections made the system more flexible, as data could be updated or retrieved easily at runtime.

Admin Interface – Dynamic Updates

The ICourierAdminService interface was implemented using collections to allow the admin to perform operations like creating employees, companies, and locations dynamically. Admins can also view or update orders and payments using the data stored in these collections.

The usage of lists for employees, companies, and locations ensured that the system could handle any number of entries without needing to worry about size constraints.

User Implementation – Dynamic Updates

In the user module, the ICourierUserService interface was implemented with ArrayList and HashMap for dynamic behavior. Users can create accounts, place courier orders, track their status, and cancel them. The courierMap collection links each user to their respective orders efficiently.

All user-related operations use the collection data passed from the main module, making the flow of data consistent and real-time.

Usage and Benefits of Collections

- ArrayList is used for storing items like users, employees, payments, locations, and courier companies. It supports dynamic resizing and gives easy access to elements.
- HashMap is ideal for mapping user IDs to their list of couriers, enabling quick lookup and update of orders.
- This design makes the system scalable and easy to maintain, without the limitations of fixed arrays.

Conclusion

By using collections throughout the Courier Management System, data handling became more efficient and flexible. All operations, whether user-side or admin-side, now support real-time updates and can scale as needed. The use of ArrayList and HashMap simplified the logic and made the system more user-friendly and maintainable.