

PasoEats Design Explanation

Encapsulation:

- All fields are private. Variables are only accessible through getter and setter methods.

Inheritance:

- Customer, Driver, and Admin extend User to reuse its class attributes (UUID id, String username, String name).

Polymorphism:

- For our command line interface we will use different fields from the User class for each user type.

Abstraction:

- User class is abstract and has abstract methods for each user type. This design enforces a common interface for all user types while abstracting away the unique implementation details of their specific actions.
- The OrderManager class exposes place(), acceptNext(), markStatus() and get() methods to the user while hiding its complex internal data structures such as Map and ArrayDeque. This approach ensures users interact only with necessary functionality, insulating them from internal changes to the storage implementation.

Data Structures:

- .byId: Map<UUID, Order> - to store orders by their ID and have O(1) lookup time.
- intake: ArrayDeque<UUID> - to store orders in the order they were placed and have O(1) insertion and removal time. FIFO (First In, First Out).
- List<OrderItem> (inside Order) - to store the items in the order.
- UUID (User, Order, Driver, Customer, Admin) - to identify each object uniquely and control access to them in a secure way.

Overall Complexity:

Time Complexity: All Methods have a time complexity of O(1) except for .contains in the ArrayDeque. Similarly the HashMap will take O(n) time for the following functions: .put .remove .get .containsKey.

Space Complexity: O(n) for the map and queue.

Work Distribution:

- Reese
 - OrderManager
 - Order
- Connor
 - User
 - Driver
 - Customer
 - Adminstrator
- Brian
 - DriverPool
 - RestaurantManager
 - MenuItem
 - *(optional, stretch goal) Graphical User Interface*
- Everybody
 - Command Line Interface

Class Diagram

