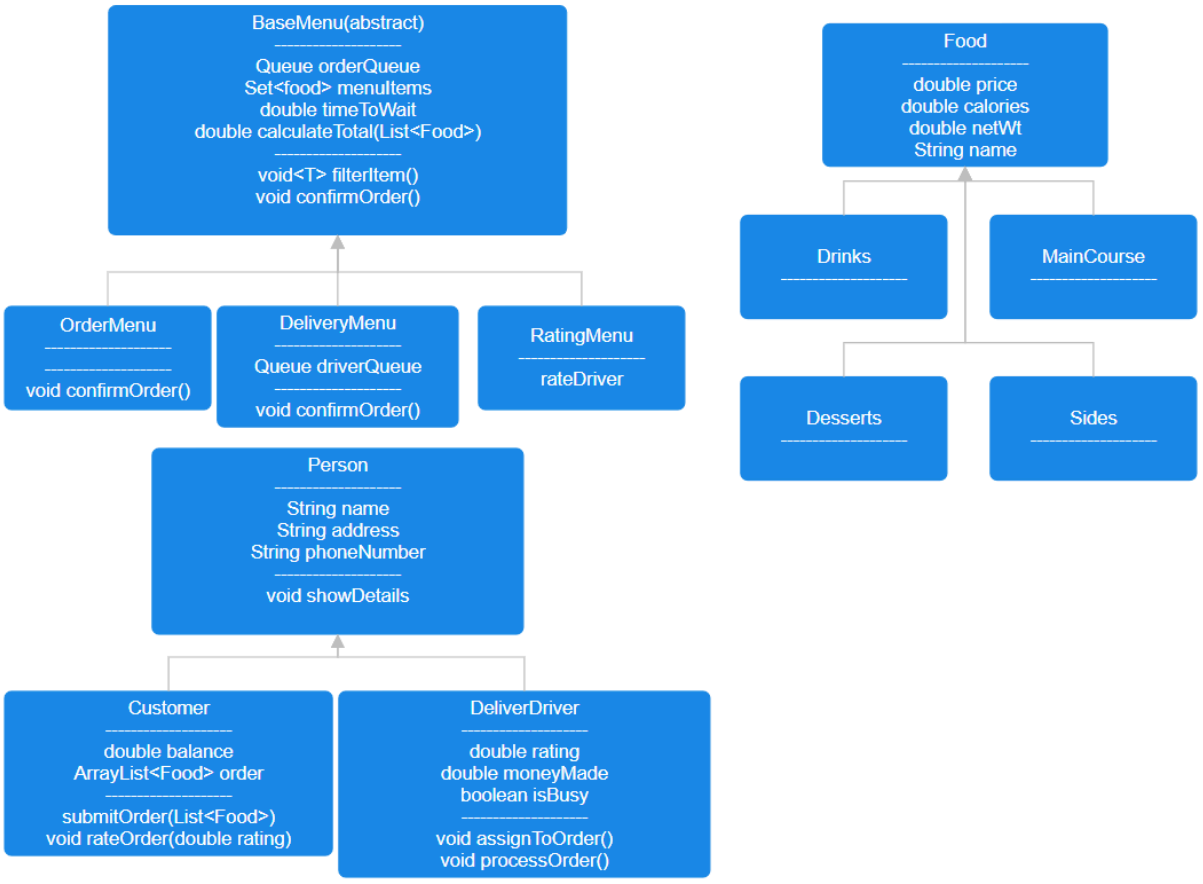

Jacob Rossel
Austin Tavares
Israel Aguiniga

This project utilizes the four main principles of OOP: **abstraction**, **polymorphism**, **inheritance**, and **encapsulation**.

1. **Abstraction**: Abstraction is achieved through the design of the BaseMenu abstract class, which defines core attributes and behaviors (such as orderQueue, menuItems, and calculateTotal) that are shared across different types of menus. This design allows us to create specialized menus, like OrderMenu, DeliveryMenu, and RatingMenu, each providing different ways for users to interact with the system. By focusing only on essential details in BaseMenu, we hide complex implementation details from the user, making the system more intuitive and flexible.
2. **Polymorphism**: Polymorphism is demonstrated in the Food superclass and its subclasses (Drinks, MainCourse, Desserts, and Sides). Each food item inherits from Food, allowing them to be treated as similar objects regardless of their specific type. This enables the system to interact with any Food item using the same methods, such as retrieving the price or name, while still maintaining distinct attributes and behaviors for each specific type (e.g., Desserts may have extra properties like "sugar content").
3. **Inheritance**: Inheritance is used extensively to simplify code and promote reusability. Common traits and methods, like confirmOrder in BaseMenu or showDetails in Person, are defined once in the superclass and inherited by all subclasses, reducing redundancy. For example, Customer and DeliverDriver both inherit from Person, which centralizes common properties like name, address, and phoneNumber. This structure allows for efficient code management, ensuring shared attributes exist among related classes while preserving class-specific functionalities.
4. **Encapsulation**: Encapsulation is applied to keep certain attributes private to each class, ensuring that data is protected and accessible only through controlled methods. For example, Customer may have a private balance and an order list that are only accessible and modifiable through specific methods within Customer. Similarly, DeliverDriver has private attributes like rating, moneyMade, and isBusy, which are managed internally to ensure data integrity and prevent unintended external modifications.

In addition to OOP principles, various **data structures** are used to manage the system's workflow efficiently:

- **Queue** in BaseMenu: The orderQueue in BaseMenu helps manage orders in a First-In, First-Out (FIFO) manner, ensuring that orders are processed in the order they were received.
- **Set** in BaseMenu: A set is used for menuItems, allowing the system to display unique menu items with associated prices. This helps avoid duplicate entries and improves retrieval efficiency.
- **Queue** in DeliveryMenu: The driverQueue helps manage the availability of delivery drivers, allowing the system to keep track of drivers who are free to take new orders and those currently delivering.
- **List** in Customer: The order list in Customer keeps track of items ordered by each customer, enabling easy management of their selections for later processing.