Methodology and OOP Features in the MerkelRex Crypto Trading Platform

Methodology Behind the Code

The MerkelRex crypto trading platform was developed as a comprehensive exercise in object-oriented programming (OOP) using C++. The project followed a structured, incremental methodology, where each major feature was built as a separate module and then integrated into the final system. The development process emphasized modeling real-world entities and their interactions, mirroring the architecture of an actual crypto exchange.

Key steps in the methodology included:

- **Data Modeling:** Identifying core entities such as wallets, orders, and users, and representing them as classes with relevant attributes and methods.
- **Incremental Implementation:** Starting with basic functionalities (like wallet management), then gradually adding features such as order placement, order matching, and transaction processing.
- **Testing and Integration:** Each component was tested individually before being integrated into the larger system, ensuring modularity and reliability.
- **Command-Line Interface:** User interaction was facilitated through a command-line interface, allowing for straightforward input/output and debugging.
- **Iterative Refinement:** The codebase was refined through multiple iterations, focusing on improving encapsulation, reducing coupling, and enhancing maintainability.

OBJECT-ORIENTED PROGRAMMING FEATURES USED

The MerkelRex project was designed to showcase and reinforce all major OOP principles and features:

OOP Feature	Implementation in MerkelRex
Classes	Core entities (e.g., Wallet, Order, User) modelled as classes
Objects	Instances of classes represent individual users, wallets, and orders
Encapsulation	Data members (like balances) are private; access via public methods, protecting internal state

OOP Feature	Implementation in MerkelRex
Abstraction	Classes expose only necessary methods (e.g., deposit, withdraw, placeOrder), hiding internal logic
Inheritance	Shared functionality (such as base order types) can be extended by derived classes for specific order types
Polymorphism	Methods can be overridden in derived classes for specialized behaviour (e.g., different order processing logic)
Static vs Non-Static	Both static (class-level) and non-static (object-level) methods are used, teaching when each is appropriate
Statefulness	Objects maintain state across operations, such as wallet balances and open orders
Association	Objects interact, such as orders updating wallet balances and vice versa

The MerkelRex platform is a practical demonstration of OOP principles in C++, with a focus on:

- Modeling real-world entities as classes and objects
- Using encapsulation and abstraction to protect and manage data
- Applying inheritance and polymorphism for code reuse and flexibility
- Managing object state and interactions to simulate a functioning crypto exchange
- This methodology not only results in a robust simulation but also provides learners with a deep, hands-on understanding of object-oriented programming in a real-world context.