**Question: Share Market Trading System – OOP, Multithreading & Exception Handling**

**Problem Statement:**

Design and implement a simplified **Share Market Trading System** using **Object-Oriented Programming** in Java that includes the following features:

**🔹 Functional Requirements:**

1. **Users and Accounts**
   * Create a class Trader with properties like traderId, name, and portfolio (list of stocks).
   * Each trader can **buy** or **sell** stocks. Buying/selling affects their portfolio.
2. **Stocks and Market**
   * Create a class Stock with attributes: stockId, stockName, pricePerShare, and availableShares.
   * Implement a class Market that holds a collection of Stock and allows traders to trade.
3. **Multithreading Requirement**
   * Simulate **concurrent trading** by multiple traders using threads.
   * Each thread represents a trader trying to **buy or sell** stocks concurrently.
   * Ensure thread-safe operations using synchronized methods or blocks to avoid race conditions when updating available shares.
4. **Exception Handling**
   * Create custom exceptions:
     + InsufficientSharesException – when trying to sell more than owned.
     + StockNotFoundException – when trading an invalid stock ID.
     + InsufficientMarketSharesException – when market doesn't have enough shares to sell.
   * Use try-catch blocks to handle exceptions gracefully during trading.
5. **Use of Interfaces and Abstraction**
   * Define a Tradeable interface with methods like buyStock() and sellStock() to be implemented by the Trader class.
   * Apply abstraction and inheritance wherever suitable.

**🔹 Additional Requirements:**

* Print a **final summary** of all trader portfolios after all trading threads complete.
* Ensure proper **exception messages** are logged during erroneous transactions.
* Use proper **class design** to reflect real-world modeling (use encapsulation, constructors, getters/setters, etc.)

**Expected Concepts to Demonstrate:**

* **OOP Principles**: Abstraction, Inheritance, Polymorphism, Encapsulation
* **Interfaces and Classes**
* **Custom Exception Handling**
* **Multithreading and Synchronization**