Pitfalls	Correction	Description
Long Methods	Break Methods into Smaller Units	Refactor backupComputerStatuses into methods like generateStatuses and writeStatusesToFile.
Conditional Complexity	Simplify Exception Handling	Create a separate method for exception handling or simplify error handling code.
Large Classes	Refactor Methods	Break down methods in CustomerManager into smaller methods with clear responsibilities.
Inappropriate Intimacy	Use Dependency Injection	Inject MongoCollection and DatabaseConnection instead of directly instantiating them.
Large Classes	Refactor into Multiple Classes	Separate responsibilities into classes like ComputerManager, TariffManager, and HistoryManager.
Long Methods	Split Methods	Divide stopComputer into methods like calculateCost, updateHistory, and notifyListeners.
High Coupling	Use Dependency Injection	Apply dependency injection or use a factory for creating panel instances.
Single Responsibility	Delegate Responsibilities	Refactor MainMenu to delegate responsibilities to specialized classes.
Literal Usage	Define Constants	Replace literal usage in createMainPanel with constants.
Single Responsibility	Refactor Responsibilities	Separate logic into more specific classes for customers, computers, and rentals.
Code Duplication	Consolidate Methods	Create a generic method or use Stream to consolidate search methods.
Literal Usage	Externalize Messages	Use constants or externalize messages instead of hardcoded strings in System.out.println.
Lack of Flexibility	Add Modifiable Methods	Implement methods to update tariffs in TariffManager.
Duplication of Responsibility	Separate Responsibilities	Divide tariff creation and access into different methods or classes.
Long Methods	Break Down Long Methods	Divide getActiveDuration() into smaller methods handling individual conditions.
Duplicate Code	Use Common Methods	Refactor duplicate code in getActiveDuration() into a common method.
Feature Envy	Move Cost Calculation	Transfer calculateCost() to the Tariff class if it aligns with tariff management.

Inappropriate Intimacy	Encapsulate State Management	Encapsulate internal state management in Computer and provide higher-level methods for interaction.
Speculative Generality	Remove Unnecessary Methods	Eliminate setActive() if it is not needed or used elsewhere.
Large Classes	Split into Smaller Classes	Refactor Customer class into smaller classes if it becomes complex.
Speculative Generality	Consolidate Attributes and Methods	Remove or consolidate unnecessary attributes and methods in the Customer class.
Uncommunicative Names	Use Descriptive Names	Replace generic names like getId() and setId() with more descriptive names.
Long Methods	Break Down Long Methods	Split customerMenu() into smaller methods handling individual responsibilities.
Conditional Complexity	Simplify Validation Logic	Refactor complex nested conditionals in customerMenu() into simpler methods.
Redundant or Meaningless Comments	Remove or Complete Comments	Eliminate unnecessary comments or complete commented-out methods like showCustomers().
Feature Envy	Decouple from CustomerManager	Reduce reliance on CustomerManager in CustomerMenu by delegating tasks to other components.
Long Methods	Split Methods	Divide constructor and getInstance() into methods handling connection setup and error handling separately.
Speculative Generality	Define Constants and Configure	Replace hardcoded CONNECTION_STRING and DATABASE_NAME with configurable settings.
Inappropriate Intimacy	Separate Database Operations	Isolate connection handling and database operations into different classes.
Feature Envy	Decouple from MongoDB-Specific Details	Abstract MongoDB-specific details to a separate class or interface.
Inappropriate Intimacy	Delegate User Input Handling	Refactor user input handling to separate classes or components.
Large Classes	Split into Smaller Components	Break down CustomerManagementPanel into smaller, more manageable classes.
Long Methods	Refactor Methods	Divide methods in CustomerManagementPanel into smaller methods handling specific tasks.

Feature Envy	Reduce Dependence on CustomerManager	Minimize direct usage of CustomerManager within CustomerManagementPanel.
Inappropriate Intimacy	Refactor to Handle Text Field Operations	Separate setNumericOnly logic into a utility class or method.
Large Class	Break Down into UI and Business Logic	Divide large class responsibilities into UI handling and business logic components.
Long Methods	Refactor into Smaller Methods	Split methods in TimerManager into smaller methods handling individual tasks.
Duplicate Code	Use Utility Methods	Create utility methods for common code used in createStyledLabel and createStyledButton.
Inappropriate Intimacy	Refactor to Manage UI and Business Logic	Separate UI management from CyberManager to maintain clear responsibilities.
Magic Numbers	Use Constants	Replace hardcoded interval values with named constants for better readability.
Hard-Coded Path	Use Configurable Paths	Externalize image path or make it configurable instead of hardcoded.