SOLID Principle	Description	Applied in the Code	Code Examples	Comments
Single Responsibility	A class should have only one reason to change, meaning it should have a single responsibility.	Partially	The CyberManagementPanel class handles multiple responsibilities.	Could be refactored to separate UI logic, timing, and computer management.
Open/Closed	Entities should be open for extension but closed for modification.	No	-	The CyberManagementPanel class is not designed to be easily extended without modifying the code.
Liskov Substitution	Objects of a derived class should be replaceable with objects of the base class without altering the functioning of the program.	Not applicable (no inheritance in the shown code)	-	Inheritance is not used in the provided code.
Interface Segregation	Clients should not be forced to depend on interfaces they do not use.	Not applicable (no interfaces in the shown code)	-	Interfaces are not used in the provided code.
Dependency Inversion	Dependencies should be on abstractions, not on concrete details. Abstractions should not depend on concrete details.	Partially	java\npublic CyberManagementPanel(CyberManager cyberManager) {\n this.cyberManager = cyberManager;\n // ...\n}\n	The CyberManagementPanel class depends directly on the concrete implementation of CyberManager. Using an interface could better adhere to this principle.