Assignment 2 Design Rationale
Requirement 4: Static Factory's staff benefits

Design Goal:
The primary objective for this assignment is to integrate a Computer Terminal within the spaceship that allows the player (intern) to interactively purchase items to aid them.

Design Decision:
We implemented the Computer Terminal as a class that extends Ground, giving it a fixed presence in the game environment. This terminal includes a variety of items that implement the Purchasable interface, ensuring a flexible and scalable system for handling different types of items. Each item has its own price and effect, enhancing the gameplay with diverse options and strategies.

Alternative Design:
An alternative approach for the Computer Terminal functionality could have been to integrate all purchasing and item interaction logic directly within the Actor class, rather than using a separate ComputerTerminal class. In this design, the Actor would carry methods to handle item purchases, check for available items, and manage transactions directly. This method would consolidate the control over transactions and inventory within a single class, potentially simplifying the overall system architecture by reducing the number of interacting components.

Analysis of Alternative Design:
The approach of placing all transaction logic directly in the Actor class has several implications:

1. Scalability and Extensibility Issues:
    - Placing all purchasing logic within the Actor class could hinder scalability. As new types of transactions or purchasable items are introduced, the Actor class would require frequent modifications, which violates the Open/Closed Principle. This could lead to an Actor class that is difficult to maintain and extend.
2. Reusability and Maintenance Challenges:
    - This design minimizes reusability since any new actor types or non-player characters that also need purchasing capabilities would either need to duplicate the transaction logic or inherit from a potentially overly complex Actor superclass. Maintenance would become challenging as changes to transaction logic might impact various unrelated parts of the Actor class.
3. Violation of Single Responsibility Principle:

- The Actor class would take on additional responsibilities, overseeing not just actor behaviors and states but also managing inventory and transactions. This violates the Single Responsibility Principle, which states that a class to have only one reason to change, keeping it concise.

Final Design:

The design involves key classes:

1. **ComputerTerminal -** Manages the availability and transaction of items. It directly supports game interaction by extending Ground, integrating into the game world's layout.
2. **Purchasable Interface -** Abstracts the purchasing behavior, allowing any item to define its own purchasing logic. This ensures flexibility and scalability in adding new purchasable items.
3. **Items (EnergyDrink, DragonSlayerSword, ToiletPaperRoll) -** Implements both Purchasable and various gameplay effects (e.g., healing, attacking), demonstrating diverse functionalities that affect the player's strategy and resource management.

Principles Applied:

1. **(S)ingle Responsibility Principle (SRP):**
   - Each class has one responsibility — ComputerTerminal handles item listing and interaction, while each item manages its own purchasing logic and effects.
2. **(O)pen/Closed Principle (OCP):**
   - The Computer Terminal can incorporate new items without modification, adhering to OCP by allowing extensions through new Purchasable implementations.
3. **(L)iskov Substitution Principle (LSP):**
   - Any item implementing Purchasable can be used interchangeably in the terminal, ensuring that changes in item types do not disrupt the terminal's operations.
4. **(I)nterface Segregation Principle (ISP):**
   - Purchasable provides a narrow interface for purchasing operations, ensuring that items are not forced to implement unnecessary methods.
5. **(D)ependency Inversion Principle (DIP):**
   - High-level modules like ComputerTerminal depend on abstractions (Purchasable), not on concretions (EnergyDrink, DragonSlayerSword, ToiletPaperRoll).

Conclusion:

The chosen design, utilizing a separate ComputerTerminal class, excels by adhering to object-oriented design principles. It supports the Single Responsibility Principle by separating transaction logic from actor behavior, and complies with the Open/Closed Principle, allowing new purchasing functionalities without modifying existing code. Thus, the decision for a modular, principle-aligned architecture ensures better maintainability, scalability, and clarity.