VOID ANOMALY

Estimated time: 40 minutes
Goal: Understand core object-oriented design concepts (inheritance, association, aggregation and composition, multiplicity, and abstraction) through UML class diagrams and requirement-based modeling.

Background Story

Amazing work on creating a Recovery Protocol Guide and stabilizing the portal! Thanks to your hard work, the portal finally activates and is ready to send you back home.

Hmmm so everything is ok now and there is no bad thing happening? It's too good to be true, isn't it? (Just like how you write a complex program and it runs on the first try with no error.) If you think so, then you're right. It's too good to be true so we're adding some random event.

As your rescue team begins crossing the dimensional gateway, a sudden anomaly wakes up, feels cute and decides to destroy the portal (just like how a bug that was already hiding in the system finally gets triggered). Thanks to this cutie patootie, your team is now trapped inside The Void Between Worlds. (That one group of friends who never seem to catch a break and always something happening to them)

The UML Blueprint that connects the dangerous valley (where you previously got teleported) to the Realm of Enchantments is now corrupted. The exit pathway is now unstable and blurry with only fragments of the system structure remaining visible. Without a correct high-level blueprint, the portal cannot reconstruct reality.

To escape, your team must analyze the system requirements, and repair the glitchy & corrupted UML class diagram so that the portal architecture follows correct object-oriented design rules.

You have 30 minutes before the Void collapses and your existence deleted permanently.

System Requirements

Functional Structure

PortalSystem:

- Represents the overall system with a name and version
- Managed by one SystemAdministrator
- Controls one or more DimensionalZones (destroying the system automatically destroys all zones and portals)
- Key actions: initializeSystem(), shutdownSystem(), add/remove DimensionalZones

DimensionalZone:

- Represents a unique dimension with a zoneID and stabilityLevel.
- Managed by one ZoneController
- Contains one or more Portals
- Key actions: stabilizeZone(), add/remove Portals

Portal:

- Represents a gateway with portalID and energyLevel
- Can transport multiple Travelers
- Key actions: activate(), deactivate(), transport()

Traveler:

- Represents a person moving through portals, with name and travelerID
- Can use multiple Portals
- Key actions: enterPortal(), exitPortal()

Staff:

- Base class for all staff members with name and employeeID.
- Subtypes: SystemAdministrator, ZoneController, PortalOperator.
- Key idea: Each staff type handles different parts of the system.

SystemAdministrator:

- Attributes: adminLevel: int (e.g., 1-5)
- Methods: createPortalSystem(), deletePortalSystem(), assignZoneController()

ZoneController:

- Attributes: controlledZones: List<DimensionalZone>
- Methods: monitorStability(), stabilizeZone()

PortalOperator:

- Attributes: operatedPortals: List<Portal>
- Methods: activatePortal(), deactivatePortal(), transportTraveler()

Design Constraints

- Destroying a PortalSystem automatically destroys all DimensionalZones and Portals inside it (specifies composition)

- Use inheritance where appropriate

- Use associations instead of duplicated attributes

- Apply correct aggregation/composition

- Use correct multiplicity

- Avoid unnecessary one-to-one relationships

- Follow object-oriented design principles

Max 30 minutes fixing the diagram, 10 mins presenting (for all groups)

In-person labs: 5-6 members in each group (depends on students present), paper or diagram tools
Online labs: 5-6 members in each group (depends on the students present), diagram tools

Corrupted UML Class Diagram