

PixelCloud Recovery Protocol

Estimated time: 30 minutes

Goal: Understand the concept of serialization, buffering, file I/O basics, closing streams and java libraries through explaining the concept to one another.

Background Story

Congratulations on successfully defending your survival camp in the dangerous valley and staying alive. After days of holding off infected creatures, a rescue team has finally arrived! Plot twist: they aren't from the Realm of Enchantments but rather from a tech company, named PixelCloud, from another universe that's been tracking portal anomalies.

The good news is they can finally help you get back home but the bad news is their rescue system is currently completely broken. This dimensional portal technology relies on a complex file management system (just because) to safely transport survivors back home, and it's failing everywhere.

To add more drama, the lead engineer just got dragged away by an infected creature. Before disappearing, they shouted loudly: "*DOCUMENTATION....FIVE CRITICAL SYSTEMS...FIX THEM IF YOU WANNA GO HOMMEEeee.....*" (Wonder why nobody tried to save him but anyways)

You and your fellow survivors have been split into five emergency repair teams. Each team must immediately create a Recovery Protocol Guide for ONE broken system. If even ONE guide is incomplete, the portal won't stabilize and NOBODY goes home.

The infected creatures will breach the camp in 30 minutes.

Can your team document the fix in time?

Your Mission

Each team creates a **one-page emergency protocol** for other survivor-engineers to fix their part of the portal system. Think of it as a field manual that works even when you're exhausted, scared, and have no cell signal.

Materials

In-person: (large) paper, sticky notes, markers, pens

Online: Miro, powerpoint, canva

Room setup

In-person: different stations around the perimeter

Virtual: different breakout rooms

The number of stations and breakout rooms depend on the students present.

The Challenge

Phase 1: Document Your System (20 mins)

The rescue portal has FIVE critical systems. Your team is assigned ONE. Create a protocol guide based on the assigned system and ensure your guide meets all the requirements. Each team can randomly draw one critical system. For online teams, different systems will be automatically assigned to each breakout room.

Phase 2: Emergency Briefing (10 mins)

2 members stay as "System Specialists" (experts who brief others)

Others will **spend around 1 minute per station** as "Cross-Training Survivors" (need to understand backup systems)

System Specialists brief, for example:

"This system controls..."

"Here's what breaks if we mess up..."

"The fix requires..."

Cross-Training Survivors can ask, for example:

"What if creatures attack mid-process?"

"Can you explain that diagram again?"

"When exactly do we activate this?"

TEAM 1: Power Grid Overload

System: Buffering

The crisis: "The portal's energy reader is processing 10,000 dimensional coordinates per second, one at a time. It's overheating and will explode before we can evacuate everyone!"

Your protocol should explain:

- Why reading coordinates one-by-one is overloading the system
- How energy buffering works (collect, then transfer in batches)
- Survivor analogy: Gathering supplies vs. making 100 trips to storage
- Snippet of code showing BufferedReader managing coordinate data
- Fatal error: Buffer full during creature attack = lost evacuation data
- Activation: Large data transfers, rapid coordinate calculations

TEAM 2: Data Type Catastrophe

System: FileInputStream vs FileReader (Bytes vs Characters)

The crisis: "The portal is reading survivor names as BYTES instead of TEXT! Everyone's name shows up as weird numbers. We can't identify who to evacuate—we might send the wrong people home!"

Your protocol should explain:

- Difference between byte streams (FileInputStream) and character streams (FileReader)
- When to use bytes (images, serialized objects, binary data) vs. characters (text, names, messages)
- Visual showing byte data vs. text data
- Code examples of BOTH stream types
- Fatal error: Using FileInputStream for survivor profiles = garbled names = wrong people evacuated
- Activation: Choose based on DATA TYPE you're reading/writing

TEAM 3: Survivor Data Corruption

System: Serialization

The crisis: "The portal tried to save survivor profiles for transport, but the files are corrupted! It can't reconstruct who to send home. People might arrive... incomplete."

Your protocol should explain:

- What survivor data serialization does (person → transportable format)
- How to make survivor profiles serializable
- The "transient" concept (some data can't cross dimensions)
- serialVersionUID (dimensional compatibility checking)
- Fatal error: Non-serializable magical artifacts crash the whole portal
- Activation: Saving complex survivor data, magical item transport

TEAM 4: Resource Leak Cascade

System: Closing Streams & Resource Management

The crisis: *"Every time we test the portal, it opens a dimensional rift but never closes it properly. Now we have 50 open rifts draining power. One more test and the system crashes!"*

Your protocol should explain:

- What happens when portals aren't closed (dimensional leaks)
- Why the system has limited rift capacity (energy constraints)
- try-with-resources auto-closing mechanism
- Visual: Available energy decreasing with each unclosed rift
- Fatal error: Opening evacuation portal in a loop without closing = total system failure
- Activation: EVERY dimensional operation—always close rifts!

TEAM 5: Abandoned Engineer's Toolkit

System: Built-in Java Libraries

The crisis: *"The engineer who disappeared was rebuilding basic tools from scratch instead of using the portal system's built-in utilities. They probably wanted to show off their skills but they wasted days! We don't have that kind of time!"*

Your protocol should explain:

- What tools exist in the system (util, time, math, text libraries)
- What each toolkit offers (quick reference diagram)
- Why use built-in vs. building new (tested under dimension stress)
- Quick/simple code samples using Java libraries
- Fatal error: Building custom dimensional calculator when Math library exists = wasted evacuation time
- Activation: Random coordinate generation, timestamp logging, coordinate math