#4156: Fix reordering issue in AzureOpenAiChatModel (Fixes #4155)

🔍 Problem Analysis

Problem Summary:

The AzureOpenAiChatModel has a race condition in streaming responses where chat chunks can arrive out of order due to Flux::flatMap not providing ordering guarantees. This causes streamed content like 'data: <A>' followed by 'data: <B>' to potentially be received as '<B> => <A>'.

Key Requirements:

- Maintain strict ordering of streaming chat response chunks
- Preserve existing Spring AI chat model API contract and behavior
- Ensure compatibility with Azure OpenAI streaming protocol
- Maintain thread-safety in reactive streaming operations
- Support concurrent processing while preserving order
- Avoid breaking changes to existing chat model implementations
- Maintain performance characteristics of streaming responses
- Handle tool execution flow without introducing new race conditions

Design Decisions:

- Replace Flux::flatMap with flatMapSequential to enforce ordering guarantees
- Apply the ordering fix to both main chat response stream and tool execution flow
- Keep the existing reactive pipeline structure intact to minimize risk
- Maintain the same merge and processing logic for chat completions
- Preserve all existing error handling and response mapping behaviors

Solution Approach:

- Primary approach: Replace flatMap with flatMapSequential to maintain emission order
- Applied consistently across both chat response processing and tool execution flows
- Minimal invasive change targeting only the specific ordering

🔄 Backport Assessment

✅ **APPROVE**

Classification: Bug Fix

Risk Level: Low - minimal code change addressing concurrency issue without affecting public interfaces

Reasoning:

This is a clear bug fix that resolves a reordering issue in Azure OpenAI chat streaming without changing any public APIs. The change is limited to internal reactive stream processing and maintains backward compatibility while fixing a concurrency-related bug.

Recommendations:

Safe for backporting. Consider testing the fix in 1.0.x environment to ensure the reactive streams behavior is consistent across Spring versions. No special migration considerations needed.