Lesson 5: Multi-Agent Orchestration with OpenTelemetry

This lesson demonstrates sequential orchestration where multiple specialized agents work together in a defined sequence to analyze a complete investment portfolio, while implementing comprehensive OpenTelemetry tracing for observability.

Prerequisites

1. **Install AI Toolkit for VS Code**:
    - Open Visual Studio Code
    - Go to Extensions (Ctrl+Shift+X)
    - Search for "AI Toolkit"
    - Install the **AI Toolkit** extension by Microsoft
    - Restart VS Code if needed
2. **Configure AI Toolkit Tracing**:
    - Open the **AI Toolkit** panel in VS Code
    - Navigate to the **Tracing** section
    - Click **Start Collector** to start the local OTLP trace collector
    - The collector will start listening on `http://localhost:4317` (gRPC) and `http://localhost:4318` (HTTP)

Getting Started

1. Switch to Lesson 5 directory:

```
cd workshop-agent-framework/dotnet/Lessons/Lesson5
```

2. Run the application to see the current basic implementation:

```
dotnet run
```

Try entering some stock symbols like "MSFT, AAPL" to see the manual sequential execution.

Implementation Steps

Open `Program.cs` and enhance it with OpenTelemetry observability and proper workflow orchestration:

Step 1: Add OpenTelemetry Imports

Add the OpenTelemetry imports at the top of the file:

```
using Microsoft.Agents.AI.Workflows;
using OpenTelemetry;
using OpenTelemetry.Trace;
```

Step 2: Configure OpenTelemetry TracerProvider

Replace the comment about TracerProvider with the actual implementation:

```
// Create TracerProvider that exports to console and OTLP
// Following Python Agent Framework pattern: uses gRPC on port 4317
using var tracerProvider = Sdk.CreateTracerProviderBuilder()
    .AddSource("agent-telemetry-source")
    .AddConsoleExporter()
    .AddOtlpExporter(options =>
    {
        // Primary: gRPC on 4317 (matches Python Agent Framework and AI
Toolkit)
        options.Endpoint = new Uri("http://localhost:4317");
        options.Protocol = OpenTelemetry.Exporter.OtlpExportProtocol.Grpc;
    })
    .Build();
```

Step 3: Add OpenTelemetry Instrumentation to Agents

Replace each agent creation with OpenTelemetry-instrumented versions:

```
// Portfolio Research Agent with OpenTelemetry
ChatClientAgent portfolioResearchAgent = chatClient
    .AsBuilder()
    .UseOpenTelemetry()
    .Build();

portfolioResearchAgent = new ChatClientAgent(
    portfolioResearchAgent,
    instructions: researchAgentInstructions,
    name: "PortfolioResearchAgent",
    description: "Gathers market data and news for portfolio stocks",
    tools: [stockPriceTool, webSearchTool, timeTool]
);

// Risk Assessment Agent with OpenTelemetry
ChatClientAgent riskAssessmentAgent = chatClient
    .AsBuilder()
    .UseOpenTelemetry()
    .Build();
```

```
riskAssessmentAgent = new ChatClientAgent(
    riskAssessmentAgent,
    instructions: riskAgentInstructions,
    name: "RiskAssessmentAgent",
    description: "Analyzes portfolio risk and diversification"
);

// Investment Advisor Agent with OpenTelemetry
ChatClientAgent investmentAdvisorAgent = chatClient
    .AsBuilder()
    .UseOpenTelemetry()
    .Build();

investmentAdvisorAgent = new ChatClientAgent(
    investmentAdvisorAgent,
    instructions: advisorAgentInstructions,
    name: "InvestmentAdvisorAgent",
    description: "Provides investment recommendations based on research and
risk analysis"
);
```

Step 4: Create Sequential Workflow

Replace the manual agent calls in the main loop with a proper sequential workflow:

```
// TODO 6: Create Sequential Workflow with all three agents
// Replace this comment with:
var workflow = SequentialAgentWorkflow.Create(
    portfolioResearchAgent,
    riskAssessmentAgent,
    investmentAdvisorAgent
);
```

Step 5: Add Performance Monitoring

Replace the manual execution with workflow execution and performance tracking:

```
// TODO 7: Add performance monitoring with Stopwatch
// Replace the manual agent calls with:
var stopwatch = System.Diagnostics.Stopwatch.StartNew();

var prompt = $"Analyze portfolio for stocks: {string.Join(", ", symbols)}.
Provide comprehensive research, risk assessment, and investment
recommendations.";

var response = await workflow.InvokeAsync(prompt);
```

```
stopwatch.Stop();

Console.WriteLine($"\n✅ Analysis completed in
{stopwatch.ElapsedMilliseconds}ms");
Console.WriteLine($"📊 Final Investment Analysis:\n{response}");
Console.WriteLine();
Console.WriteLine("🔍 Check console output above for OpenTelemetry traces");
Console.WriteLine("💡 Open AI Toolkit in VS Code to visualize traces");
```

Step 6: Update Console Output

Add enhanced telemetry information to the console output:

```
Console.WriteLine("=== Investment Portfolio Analyzer with Sequential
Orchestration & OpenTelemetry ===");
Console.WriteLine("This demonstrates MAF Sequential Orchestration with
comprehensive telemetry:");
Console.WriteLine("  • Three specialized agents with distributed tracing");
Console.WriteLine("  • OpenTelemetry traces with console and OTLP export");
Console.WriteLine("  • Microsoft Agent Framework instrumentation");
Console.WriteLine("  • AI Toolkit for VS Code integration");
Console.WriteLine();
Console.WriteLine("🔍 OpenTelemetry TracerProvider configured with:");
Console.WriteLine("   • Console Exporter: ✅ Enabled");
Console.WriteLine("   • OTLP Exporter: ✅ http://localhost:4317 (gRPC)");
Console.WriteLine("   • Protocol: gRPC (matching Python Agent Framework)");
Console.WriteLine("   • AI Toolkit: Use tracing view in VS Code");
```

Testing with OpenTelemetry

1. **Ensure AI Toolkit is running**:
   - Check that the OTLP collector is started in VS Code AI Toolkit
   - Verify it's listening on localhost:4317
2. **Run the enhanced application**:

   ```
   dotnet run
   ```

3. **Test with portfolio analysis**:
   - Enter stock symbols: `MSFT, AAPL, NVDA`
   - Observe comprehensive telemetry output in console
   - Watch for OpenTelemetry activity traces with timing information
4. **View traces in AI Toolkit**:

- In VS Code, open AI Toolkit panel
- Navigate to Tracing section
- Click "Refresh" to see new traces
- Select a trace to explore the span tree
- Review Input + Output tab for AI message flows
- Check Metadata tab for detailed trace information

Expected Behavior with OpenTelemetry

You should observe:

1. **Sequential Agent Execution**:
   - Portfolio Research Agent gathers market data and news
   - Risk Assessment Agent analyzes portfolio composition and risk
   - Investment Advisor Agent provides recommendations
2. **Comprehensive Telemetry Data**:
   - Console output shows OpenTelemetry activities with IDs and timing
   - Each agent's execution is traced with performance metrics
   - Complete workflow orchestration visibility
3. **AI Toolkit Integration**:
   - Traces appear in VS Code AI Toolkit tracing view
   - Span hierarchy shows workflow → individual agents
   - Detailed metadata about model calls, token usage, and response IDs
4. **Performance Monitoring**:
   - Total execution time displayed
   - Individual agent timing visible in traces
   - Memory and resource usage tracking

This enhanced lesson demonstrates how to build production-ready multi-agent systems with enterprise-grade observability using OpenTelemetry and the AI Toolkit for VS Code.