Sanidhaya's Core Agent System - Complete File Structure

Week-by-Week Development Plan with File Purposes

Project Structure Overview

```
SwarmCoordination/
├── src/main/java/com/team6/swarm/
│   ├── core/           # Your main package (Sanidhaya)
│   ├── communication/  # John's package
│   ├── intelligence/   # Lauren's package
│   └── ui/            # Anthony's package
├── src/test/java/     # Unit tests
├── resources/         # Configuration files
└── docs/             # Documentation
```

Week 1: Foundation Files

Purpose of Week 1: Basic Data Structures & Agent Creation

Goal: Create the fundamental building blocks that everyone will use

File 1: Point2D.java

- **Package:** `com.team6.swarm.core`
- **Purpose:** Represents X,Y coordinates in 2D space
- **Why needed:** Every agent has a position, every target has coordinates
- **Used by:** All team members for positions and locations
- **Key responsibility:** Store coordinates, calculate distances between points

File 2: Vector2D.java

- **Package:** `com.team6.swarm.core`
- **Purpose:** Represents direction and magnitude (like velocity or force)
- **Why needed:** Agents move with velocity, Lauren calculates movement forces
- **Used by:** Physics calculations, flocking algorithms, movement commands
- **Key responsibility:** Vector math operations (add, multiply, normalize)

File 3: AgentStatus.java (enum)

- **Package:** `com.team6.swarm.core`
- **Purpose:** Define all possible states an agent can be in
- **Why needed:** System needs to track if agents are working, failed, low battery
- **Used by:** All components to check agent health

- **Values:** ACTIVE, INACTIVE, FAILED, BATTERY_LOW, MAINTENANCE

File 4: AgentState.java

- **Package:** `com.team6.swarm.core`
- **Purpose:** Complete snapshot of an agent at any moment
- **Why needed:** This is the "passport" of each agent - contains all info others need
- **Used by:** Everyone needs to know agent position, status, capabilities
- **Key responsibility:** Store agent ID, position, velocity, battery, status, capabilities

File 5: Agent.java

- **Package:** `com.team6.swarm.core`
- **Purpose:** The actual agent - can move, receive commands, update itself
- **Why needed:** This is your main "robot" - it executes Lauren's movement commands
- **Used by:** You create them, Lauren commands them, John gets their positions
- **Key responsibility:** Update position every frame, process movement commands, manage battery

File 6: AgentManager.java

- **Package:** `com.team6.swarm.core`
- **Purpose:** Factory and registry for all agents - creates, tracks, removes agents
- **Why needed:** Someone needs to manage the swarm (create/destroy agents)
- **Used by:** Anthony sends spawn commands here, everyone queries for agent lists
- **Key responsibility:** Create agents on command, track all agents, provide agent lists

File 7: SimpleTest.java

- **Package:** `com.team6.swarm.core`
- **Purpose:** Test your basic agent system works
- **Why needed:** Verify agents can be created and move before integrating with others
- **Used by:** Just you for testing
- **Key responsibility:** Create 3 agents, make them move, verify it works

Week 1 Success Criteria:

- Run SimpleTest and see 3 agents moving around
- Agents have positions that change over time
- No crashes, basic movement works

Week 2: Communication Integration

Purpose of Week 2: Connect with John's Communication System

Goal: Your agents can share position information with John's network

File 8: AgentStateUpdate.java

- **Package:** `com.team6.swarm.core`
- **Purpose:** Message format to tell John when agent positions change
- **Why needed:** John needs to know where agents are to calculate communication range
- **Used by:** You send these to John every frame
- **Key responsibility:** Package agent position changes for transmission

File 9: CommunicationEvent.java

- **Package:** `com.team6.swarm.core`
- **Purpose:** Message format to receive notifications from John
- **Why needed:** John tells you when agents receive messages from others
- **Used by:** John sends these to you when messages are delivered
- **Key responsibility:** Notify you that an agent got a message

File 10: EventBus.java

- **Package:** `com.team6.swarm.core`
- **Purpose:** Simple system to send messages between components
- **Why needed:** You and John need to exchange information without tight coupling
- **Used by:** All components to communicate with each other
- **Key responsibility:** Publish events (like position updates) and deliver to subscribers

File 11: SystemController.java

- **Package:** `com.team6.swarm.core`
- **Purpose:** Coordinates the entire system - starts/stops everything
- **Why needed:** Someone needs to manage the overall system lifecycle
- **Used by:** Anthony's UI to start/stop simulation, coordinate components
- **Key responsibility:** Initialize all components, manage system state, coordinate shutdown

File 12: IntegrationTest.java

- **Package:** `com.team6.swarm.core`
- **Purpose:** Test integration with John's mock communication system
- **Why needed:** Verify data flows correctly between your system and John's
- **Used by:** You and John together for integration testing
- **Key responsibility:** Create agents, mock John's system, verify position sharing

Week 2 Success Criteria:

- Your agents publish position updates through EventBus
- Mock communication system receives position data
- John can successfully integrate with your position updates

Week 3: Movement Commands Integration

Purpose of Week 3: Accept Commands from Lauren's Intelligence System

Goal: Lauren can tell your agents how to move and behave

File 13: MovementCommand.java

- **Package:** `com.team6.swarm.core`
- **Purpose:** Standardized format for Lauren to give movement instructions
- **Why needed:** Lauren's flocking algorithms need to control agent movement
- **Used by:** Lauren sends these, your agents execute them
- **Key responsibility:** Define standard format for movement instructions

File 14: MovementType.java (enum)

- **Package:** `com.team6.swarm.core`
- **Purpose:** All possible types of movement commands
- **Why needed:** Different movement behaviors need different parameters
- **Values:** FLOCKING_BEHAVIOR, FORMATION_POSITION, MOVE_TO_TARGET, AVOID_OBSTACLE
- **Key responsibility:** Categorize different movement strategies

File 15: CommandPriority.java (enum)

- **Package:** `com.team6.swarm.core`
- **Purpose:** Define urgency levels for commands
- **Why needed:** Emergency avoidance should override normal flocking
- **Values:** EMERGENCY, HIGH, NORMAL, LOW
- **Key responsibility:** Ensure critical commands execute first

File 16: PhysicsEngine.java

- **Package:** `com.team6.swarm.core`
- **Purpose:** Handles realistic movement, collisions, boundaries
- **Why needed:** Agents should move smoothly and realistically
- **Used by:** Your Agent class for movement calculations
- **Key responsibility:** Calculate physics, handle collisions, enforce boundaries

File 17: TaskCompletionReport.java

- **Package:** `com.team6.swarm.core`
- **Purpose:** Tell Lauren when agents complete or fail tasks
- **Why needed:** Lauren needs feedback to make better decisions
- **Used by:** You send to Lauren, she uses for task assignment
- **Key responsibility:** Report success/failure of assigned tasks

Week 3 Success Criteria:

- Agents can receive and execute movement commands
- Basic flocking forces applied to agent movement
- Task completion reporting works

Week 4: User Interface Integration

Purpose of Week 4: Connect with Anthony's User Interface

Goal: Anthony can display your agents and send user commands

File 18: SystemCommand.java

- **Package:** `com.team6.swarm.core`
- **Purpose:** Standardized format for Anthony to send user commands
- **Why needed:** User clicks need to become agent actions
- **Used by:** Anthony sends these when user interacts
- **Key responsibility:** Define format for user-initiated commands

File 19: CommandType.java (enum)

- **Package:** `com.team6.swarm.core`
- **Purpose:** All possible user commands
- **Values:** SPAWN_AGENT, REMOVE_AGENT, SET_BOUNDARIES, START_SIMULATION
- **Why needed:** Different user actions need different handling
- **Key responsibility:** Categorize user interaction types

File 20: VisualizationUpdate.java

- **Package:** `com.team6.swarm.core`
- **Purpose:** Complete system state for Anthony's display
- **Why needed:** Anthony needs all agent data for smooth visualization
- **Used by:** You send to Anthony 30 times per second
- **Key responsibility:** Package all display data efficiently

File 21: SystemMetrics.java

- **Package:** `com.team6.swarm.core`
- **Purpose:** Performance information (CPU, memory, FPS)
- **Why needed:** Monitor system health and performance
- **Used by:** Anthony displays in UI, you calculate metrics
- **Key responsibility:** Track and report system performance

File 22: SystemEvent.java

- **Package:** `com.team6.swarm.core`
- **Purpose:** Notable events for display (agent created, failed, etc.)

- **Why needed:** Users want to see what's happening in the system
- **Used by:** All components report events, Anthony displays them
- **Key responsibility:** Standardized event reporting format

Week 4 Success Criteria:

- Anthony can spawn/remove agents through UI
- Real-time visualization shows agent positions
- System metrics displayed correctly

Week 5-6: Advanced Features

Purpose: Add Sophistication and Polish

Goal: More realistic behavior and better performance

File 23: AgentCapabilities.java

- **Package:** `com.team6.swarm.core`
- **Purpose:** Detailed information about what each agent can currently do
- **Why needed:** Lauren needs to make smart task assignments
- **Used by:** You calculate, Lauren uses for decision making
- **Key responsibility:** Report agent performance and limitations

File 24: Task.java

- **Package:** `com.team6.swarm.core`
- **Purpose:** Represents assigned work for agents
- **Why needed:** Agents need to know what they're supposed to do
- **Used by:** Lauren creates tasks, you execute them
- **Key responsibility:** Define work assignments and parameters

File 25: BoundaryManager.java

- **Package:** `com.team6.swarm.core`
- **Purpose:** Manages world boundaries and safe zones
- **Why needed:** Agents need boundaries, users may define restricted areas
- **Used by:** Physics engine for boundary checks
- **Key responsibility:** Define and enforce movement boundaries

File 26: PerformanceMonitor.java

- **Package:** `com.team6.swarm.core`
- **Purpose:** Track system performance and optimize
- **Why needed:** System must run smoothly with 10+ agents
- **Used by:** System controller for performance management
- **Key responsibility:** Monitor FPS, memory usage, optimize performance

---

Week 7-8: Formation and Coordination

Purpose: Support Complex Coordination Behaviors

Goal: Agents can form formations and coordinate complex maneuvers

File 27: Formation.java

- **Package:** `com.team6.swarm.core`
- **Purpose:** Represents a coordinated arrangement of agents
- **Why needed:** Lauren needs to command formation flying
- **Used by:** Lauren defines formations, you execute positions
- **Key responsibility:** Define formation shapes and positions

File 28: FormationType.java (enum)

- **Package:** `com.team6.swarm.core`
- **Purpose:** Standard formation patterns
- **Values:** LINE, WEDGE, CIRCLE, COLUMN, DIAMOND
- **Why needed:** Standardize formation commands
- **Key responsibility:** Define available formation patterns

File 29: CoordinationManager.java

- **Package:** `com.team6.swarm.core`
- **Purpose:** Manages multi-agent coordination
- **Why needed:** Complex maneuvers require coordination between agents
- **Used by:** Coordinates formation changes and group movements
- **Key responsibility:** Synchronize agent actions for group behaviors

---

Week 9-10: Fault Tolerance

Purpose: Handle Failures Gracefully

Goal: System continues working when agents fail

File 30: FailureDetector.java

- **Package:** `com.team6.swarm.core`
- **Purpose:** Detects when agents fail or become unresponsive
- **Why needed:** System must detect and respond to failures
- **Used by:** Monitors agent health, reports failures
- **Key responsibility:** Detect various failure modes

File 31: FailureType.java (enum)

- **Package:** `com.team6.swarm.core`
- **Purpose:** Categories of agent failures
- **Values:** SYSTEM_ERROR, BATTERY_DEPLETED, COMMUNICATION_LOST, COLLISION
- **Why needed:** Different failures need different responses
- **Key responsibility:** Classify failure types

File 32: RecoveryManager.java

- **Package:** `com.team6.swarm.core`
- **Purpose:** Handles system recovery from failures
- **Why needed:** System should gracefully handle agent losses
- **Used by:** Automatic recovery from agent failures
- **Key responsibility:** Implement recovery strategies

Week 11-12: Optional Hardware Integration

Purpose: Support Physical RC Cars (Optional)

Goal: System can control real hardware if desired

File 33: HardwareInterface.java

- **Package:** `com.team6.swarm.core`
- **Purpose:** Abstract interface for hardware control
- **Why needed:** Support both simulation and real hardware
- **Used by:** Switch between simulated and real agents
- **Key responsibility:** Unified interface for hardware/simulation

File 34: SimulationAdapter.java

- **Package:** `com.team6.swarm.core`
- **Purpose:** Implements hardware interface for simulation
- **Why needed:** Default implementation for testing
- **Used by:** Standard simulation mode
- **Key responsibility:** Simulate hardware behavior

File 35: RCCarAdapter.java (Optional)

- **Package:** `com.team6.swarm.core`
- **Purpose:** Implements hardware interface for real RC cars
- **Why needed:** Control actual hardware if available
- **Used by:** Physical demonstration mode
- **Key responsibility:** Translate commands to hardware

Week 13-14: Final Polish

Purpose: Production Ready System

Goal: Professional quality, well-tested, documented

File 36: ConfigurationManager.java

- **Package:** `com.team6.swarm.core`
- **Purpose:** Load and manage system configuration
- **Why needed:** Easy parameter tuning without code changes
- **Used by:** All components for configuration
- **Key responsibility:** Centralized configuration management

File 37: Comprehensive Test Suite

- **Package:** `com.team6.swarm.core`
- **Purpose:** Complete testing of all functionality
- **Why needed:** Ensure system works reliably
- **Files:** AgentTest.java, PhysicsTest.java, IntegrationTest.java
- **Key responsibility:** Verify all functionality works correctly

---

File Dependencies and Relationships

Core Dependencies (Week 1):

```
Point2D ← AgentState ← Agent ← AgentManager
Vector2D ← AgentState ← Agent
AgentStatus ← AgentState
```

Communication Layer (Week 2):

```
AgentStateUpdate → EventBus → John's Communication System
CommunicationEvent ← EventBus ← John's Communication System
```

Command Layer (Week 3):

```
MovementCommand → Agent → PhysicsEngine
TaskCompletionReport → Lauren's Intelligence System
```

UI Layer (Week 4):

```
SystemCommand ← Anthony's UI
VisualizationUpdate → Anthony's UI
SystemMetrics → Anthony's UI
```

Key Design Principles for Your Files

1. Single Responsibility

- Each class has one clear job
- Agent handles agent behavior, AgentManager handles agent collection
- PhysicsEngine handles movement, EventBus handles communication

2. Clear Interfaces

- Public methods are what other components use
- Private methods are your internal implementation
- Data classes (like AgentState) are mostly public fields for easy access

3. Thread Safety

- Use ConcurrentHashMap for agent collections
- AgentState should be immutable or carefully synchronized
- EventBus needs to handle concurrent access

4. Performance Oriented

- Update loops run 30-60 times per second
- Avoid creating objects in update loops
- Use efficient data structures for frequent operations

5. Integration Ready

- Clear data formats for other team members
- Event-driven architecture for loose coupling
- Mock implementations for independent testing

What Each Week Accomplishes

Week 1: You can create agents and make them move

Week 2: John can get agent positions for communication

Week 3: Lauren can command agent movement

Week 4: Anthony can display everything and handle user input

Week 5-6: More sophisticated behaviors and better performance

Week 7-8: Complex coordination and formation flying

Week 9-10: System handles failures gracefully

Week 11-12: Optional hardware support

Week 13-14: Production quality polish and testing

This structure ensures you build incrementally, integrate frequently, and always have a working system that gets progressively more sophisticated.

File Structure and Development Plan for Sanidhaya's Core Agent System
Team 6 - Distributed Multi-Agent System
Software Engineering Graduate Project