1. **技术选型**

- **编程语言**：Java
    - **原因**：Java 具有强大的库支持，适合处理复杂的逻辑和数据结构。
- **数据结构**：
    - 使用 ArrayList 存储传感器数据和计算结果。
    - 使用 HashMap 存储最终的统计结果。
- **数据处理**：
    - 对传感器数据进行初步加工（如处理负数角度）。
    - 对数据进行排序和统计分析。

2. **架构设计**

- **模块化设计**：
    - **Sensor 类**：封装传感器数据。
    - **动作类**（如 Flexion、Extension、Abduction 等）：每个类负责计算一种动作的运动幅度。
    - **Analysis 类**：负责数据的存储、计算和统计分析。
- **数据流**：
    1. 输入数据（54 个数据点）。
    2. 创建 Sensor 对象。
    3. 调用动作类的方法，计算每个动作的运动幅度。
    4. 将结果存储到对应的 ArrayList 中。
    5. 调用 Statistic 方法，对存储的数据进行统计分析，返回最终结果。

3. **质量保证**

- **测试策略**：
    - **单元测试**：为每个动作类的方法编写单元测试，确保其正确性。
    - **集成测试**：测试 Calculate 和 Statistic 方法的协同工作。
    - **系统测试**：测试整个系统的功能，确保输入数据能够正确处理并返回预期结果。

- **代码质量管理**:
    - 遵循 Java 代码规范，确保代码的可读性和可维护性。

**补充说明**

1. **数据预处理**:
    - 在 ExRotation 和 InRotation 类中，对负角度进行了处理（if(a < 0) a += 360）。
2. **扩展性**:
    - 当前代码结构清晰，便于扩展新的动作类型或传感器数据处理逻辑。