Keeping Track of Items in a Simulation

Scenario

True story: A colleague of yours is working on a simulator for robotics research, and needs to maintain a log of the information of several different types of items in the simulations that get run. Your colleague's solution was to make all the attributes of each object public, and access the data directly. Knowing this was a poor practice (but real easy, straightforward solution), your colleague asked your opinion on how to do this task better. In addition, it would be desirable to be able to indicate which objects are to be logged at run-time, rather than recompiling every time a new set of objects need to be logged. Finally, different types of objects may be added in the future--can your solution account for these?

Problem

You need to determine an easy way to log the data of objects after every time step in the simulator. At the moment, there are three types (classes) of objects in the simulator--Robots, Walls and Arenas. When a simulation is run, an instance of your Logger object will be created. The simulator will call the Logger's *logData* method every time step.

Included are four classes: Arena, Robot, Wall and Logger. You may create some interface or abstract superclass for the Arena, Robot and Wall for your solution, and add a method or two to each of these classes, but the existing functionality is not to be modified. This included changing the accessibility of the existing attributed. You are free to implement Logger as you see fit, including adding methods or attributes, but it must implement the method *logData()*, which is included in the skeleton code. For simplicity, logging should simply include printing the data of each object to the console (e.g., "Robot instance #5 has a position of (5, 2).").

Deliverables

1. Identify the design pattern you used to solve this problem, and the participants (i.e., the roles each class takes).
2. An implementation in a language of your choice.
3. A class diagram of your solution (including existing classes), so future developers can easily see how to work with your solution.