The specifications and requirements extracted from the formal partial specification in Z allow us to extract the following use cases  that cover the primary functionalities of the system. The two use cases involve deploying sensors to locations and reading temperatures. They also consider various error handling scenarios. Based on these use cases, one can identify conceptual classes, their attributes, and relationships among entities in the domain model.

**Use Case 1: DeploySensor**

Description: This use case describes the process of deploying a sensor at a specific location to monitor temperature. It includes error handling for cases where the sensor is already deployed or the location is already covered.

Primary Actors: User (or System Administrator)

Preconditions:

The system should maintain a registry of sensors and a registry of locations.

The system maintains two tables: one with already-deployed individual sensors and their respective mapped locations, and the other with those sensors and their respective read temperature.

The sensor to be deployed is not already in the table of sensor-location pairs.

The location to which the sensor will be deployed is not already associated with a deployed sensor.

Postconditions:

If the deployment is successful, the sensor-location pair is added to the sensor-location table and the sensor-temperature pair is added to the sensor-temperature table, and a message “ok” should be displayed.

If the deployment is unsuccessful because the sensor is already deployed, a message “sensor deployed” should be displayed.

If the deployment is unsuccessful because the location is already covered, a message “location already covered” should be displayed..

Flow of Events:

The user (or system administrator) initiates the deployment process and selects a sensor and a location (from the registries) to which the sensor will be mapped.

The system checks if the sensor to be deployed is not already in the set of deployed sensors, and if the location is not already covered by another deployed sensor.

If both checks pass, the system deploys the sensor and updates the set of deployed sensors, the sensor-location and sensor-temperature tables accordingly, and a success message “ok” is displayed.

If the sensor is already deployed or the location is already covered, an error message “Sensor deployed” or “Location already covered” is displayed respectively.

**Use Case 2: ReadTemperature**

Description: This use case describes the process of reading the temperature corresponding to a location. It includes error handling for cases where the location is not covered by any sensor.

Primary Actors: User (or System Administrator)

Preconditions:

The system should maintain a registry of sensors and a registry of locations.

The system maintains two tables: one with already-deployed individual sensors and their respective mapped locations, and the other with those sensors and their respective read temperature.

The requested location is covered by a sensor and the sensor-location pair is in the sensor-location table.

Postconditions:

If the location selected is covered by a sensor, the system returns the temperature corresponding to the deployed sensor at the requested location and a message “ok” is displayed.

If the location selected is not covered, the system displays a message “Location not covered”.

Flow of Events:

The user (or system administrator) initiates the temperature reading process by selecting a location.

The system checks if the requested location is in the sensor-location table.

If the location is found in that table (i.e., it is covered by a sensor), the system retrieves the temperature corresponding to that sensor and location pair from the sensor-temperature table and provides it as output and a success message “ok” is displayed.

If the location is not covered by any sensor (i.e., it is not in the sensor-location table), an error message “Location not covered” is displayed.

The following domain model was derived (drawn on draw.io):

A diagram of a flowchart

Description automatically generated

Figure 1: Domain Model