Coding Challenge

In this exercise, you will build the core of an audit system that is capable of determining the difference between two objects of the same type. The system should be capable of handling null values on both the previous and current object. Use Java or Kotlin.

Requirements:
1. If a property is updated, the system should track the name of the property, the previous value, and the current value.
2. If the property is nested, the property name should use dot notation to indicate the full path.
3. If items are added or removed from a list, the system should track the name of the property, the list of items that were added, and the list of items that were removed
4. If the property was updated on an object within a list, the property name should include square brackets with the id of the list item.
    1. The id of a list item must be a field annotated with @AuditKey or have the name 'id'. If no field meets this requirement, throw an exception that indicates that the audit system lacks the information it needs to determine what has changed.

Examples:
1. {"property": "firstName", "previous": "James", "current": "Jim"}
2. {"property": "subscription.status", "previous": "ACTIVE", "current": "EXPIRED"}
3. {"property": "services", "added": ["Oil Change"], "removed": ["Interior/Exterior Wash"]}
4. {"property": "vehicles[v_1].displayName", "previous": "My Car", "current": "23 Ferrari 296 GTS"}

What to Submit:
- The class should be named DiffTool and have a single method `diff` that takes the previous and current state of an object as arguments. The return type should be a List<ChangeType> where ChangeType is an interface/sealed class with 2 implementations: PropertyUpdate and ListUpdate.
- A README.md that describes how the DiffTool works and how long it took you to put together