DAO testing exercises

Designers of the `EmployeeProjects` database are enhancing it with timesheet tracking. They've developed a new DAO to handle creating, reading, updating, and deleting records from the `timesheet` table. For this exercise, you'll be responsible for writing DAO tests and using them to identify bugs in this new DAO. You'll then fix the bugs you've found.

Learning objectives

After completing this exercise, you'll understand:

- How to write DAO tests.
- How to use tests to find bugs in a DAO.

Evaluation criteria and functional requirements

The instruction team evaluates your code for this assignment based on the following criteria:

- The project must not have any build errors.
- You must fill out the provided `BugReport.txt` file for four bugs you found and fixed.
- Each provided DAO test method must be complete and passing.
- Code is clean, concise, and readable.

Getting started

1. You'll use the same `EmployeeProjects` database you used for the DAO exercises.
2. Import the DAO Testing exercises project into IntelliJ.

Step One: Explore starting code

Before you begin, review the provided classes in the `model` and `dao` packages.

Also, familiarize yourself with the provided test classes and the `test-data.sql` file.

Step Two: Implement the `JdbcTimesheetDaoTests` methods

In the nine test methods, replace the `Assert.fail()` with the code necessary to implement the test described by the method name. You can refer to the comments in the `TimesheetDao` interface for descriptions of what each DAO method does.

Use this unit's reading and the DAO tests from the previous DAO exercises as examples to reference while working. Static constant `Timesheet`s have been provided that you can use in your tests.

When fully implemented, five of the tests pass, and four continue to fail due to bugs in `JdbcTimesheetDao`.

Step Three: Complete bug reports and fix bugs

Fill out `BugReport.txt` with information about the four bugs you've identified in `JdbcTimesheetDao` using the DAO tests.

An example of reporting and fixing a bug

Consider the following `deleteTimesheet()` method:

```
    public void deleteTimesheet(int timesheetId) {
        String sql = "DELETE FROM timesheet WHERE timesheet_id = ?";
        jdbcTemplate.update(sql, 1);
    }
```

A method written this way would contain a bug. It always deletes the record with a `timesheet_id` of 1 rather than using the value of `timesheetId`.

There are several ways this could cause the `deleted_timesheet_cant_be_retrieved` test to fail—for example, if the test called `deleteTimesheet(2)` and then found that `getTimesheet(2)` still retrieved the timesheet.

After that test fails, you'd fix the `deleteTimesheet()` method like this:

```
public void deleteTimesheet(int timesheetId) {
    String sql = "DELETE FROM timesheet WHERE timesheet_id = ?";
    jdbcTemplate.update(sql, timesheetId);
}
```

Then you'd complete the bug report as follows:

```
Test that demonstrates problem:
    deleted_timesheet_cant_be_retrieved
Expected output:
    getTimesheet(2) returns null after calling deleteTimesheet(2)
Actual output:
    getTimesheet(2) was still returning a Timesheet object
How did you fix this bug?
    Replaced hardcoded value of 1 in deleteTimesheet with timesheetId so it
doesn't always delete the same timesheet.
```

After you've found and fixed the four bugs, all nine of the tests in `JdbcTimesheetDaoTests` pass.