5 Decision Structures

In this exercise we will test the movement of guests using decision structures (Binder Chapter 6, p123-127) and extend JPacman with moving monsters.

Exercise 30 Create a decision table following the style of Table 6.1 (p. 125) indicating what should happen when a guest tries to occupy a new cell. Cases to be distinguished include whether or not the move remains within the borders, and whether or not the move is possible based on the type of the mover (player, monster), and the type of the (optional) guest occupying the other cell.

Exercise 31 Run the current test suite and describe the coverage of the `Move`, `PlayerMove`, `Guest` and all `Guest` subclasses.

Exercise 32 Implement all entries in the decision table concerning player movements as JUnit test cases in the PlayerMoveTest class. Since the player movement has been implemented already, start by testing these.

Exercise 33 Re-run with coverage enabled, and re-assess the coverage.

Exercise 34 Explain the interplay between the abstract methods `Guest.meetPlayer` and `Move.tryMoveToGuest` and their implementations in the `Guest` and `Move` subclasses.

Exercise 35 Implement a monster move in the same style as a player move. Add a `MonsterMove` class, place it correctly in the inheritance hierarchy, and implement the required methods. Make sure you add or update appropriate invariants as well as pre- and postconditions where ever possible, and implement them using assertions.

Exercise 36 Introduce a `MonsterMoveTest` class to implement the test cases related to monster moves. You will probably want to extend `MoveTest` for this. Make sure you also update the `TestAll` class to actually invoke these tests as well. Verify your coverage using Emma.

Exercise 37 How many tests in your decision table would you need to get 100% coverage of the relevant moving methods? Why do you nevertheless need the remaining test cases?