UNIVERSITY OF ANTWERP
DEPARTMENT OF MATHEMATICS AND COMPUTER SCIENCE

# Software Testing Lab

# Assignment 4

## Submission Deadline: **March 23rd, 23:59**

### 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 1.** Create a decision table following the style of Table 6.1 (Binder p125) 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 moved object (player or monster), and the type of the (optional) guest occupying the other cell. **(Required, 10 points)**

| | Condition Section | | Action Section | | |
|---|---|---|---|---|---|
| Variant | Number of Claims | Insured Age | Premium Increase ($) | Send Warning | Cancel |
| 1 | 0 | 25 or younger | 50 | No | No |
| 2 | 0 | 26 or older | 25 | No | No |
| 3 | 1 | 25 or younger | 100 | Yes | No |
| 4 | 1 | 26 or older | 50 | No | No |
| 5 | 2 to 4 | 25 or younger | 400 | Yes | No |
| 6 | 2 or 4 | 26 or older | 200 | Yes | No |
| 7 | 5 or more | Any | 0 | No | Yes |

| | Decision Variable | Condition | Variant 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|---|---|
| **Condition Section** | Number of claims | 0 | T | T | F | F | F | F | F |
| | | 1 | F | F | T | T | F | F | F |
| | | 2–4 | F | F | F | F | T | T | F |
| | | 5 + | F | F | F | F | F | F | T |
| | Insured age | 25 or younger | T | F | T | F | T | F | DC |
| | | 26 or older | F | T | F | T | F | T | DC |
| **Action Section** | Premium Increase = 0 | | F | F | F | F | F | F | T |
| | Premium increase = 25 | | F | T | F | F | F | F | F |
| | Premium increase = 50 | | T | F | F | T | F | F | F |
| | Premium increase = 100 | | F | F | T | F | F | F | F |
| | Premium increase = 200 | | F | F | F | F | F | T | F |
| | Premium increase = 400 | | F | F | F | F | T | F | F |
| | Send warning | | F | F | T | F | T | T | F |
| | Cancel | | F | F | F | F | F | F | T |

- **Exercise 2.** Create the same table in truth table format, similar to Table 6.3 (Binder p128). **(Optional)**

- **Exercise 3.** Run the current test suite and describe the coverage of `Move`, `PlayerMove`, `Guest`, and all `Guest` subclasses. **(Required, 10 points)**

- **Exercise 4.** Implement all entries in the decision table concerning player movements as JUnit test cases in `PlayerMoveTest` class. Since the player movement has been implemented already, start by testing these. **(Required, 10 points)**

- **Exercise 5.** Re-run with coverage enabled, and re-assess the coverage. **(Required, 10 points)**

- **Exercise 6.** Explain the interplay between the abstract methods `Guest.meetPlayer` and `Move.tryMoveToGuest` and their implementations in `Guest` and `Move` subclasses. **(Required, 10 points)**

- **Exercise 7.** 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 post-conditions wherever possible, and implement them using assertions. **(Required, 20 points)**

- **Exercise 8.** Introduce a `MonsterMoveTest` class to implement the test cases related to monster moves. You will probably want to extend `MoveTest` for this. Verify the test coverage for this class. **(Required, 20 points)**

- **Exercise 9.** How many tests in your decision table would you need to get 100% coverage of the relevant moving methods? Why do you need the remaining test cases? **(Required, 10 points)**

*Late fee: -25 points*