Łamigłówka architekta

Podział na pakiety

Całość kodu została podzielona na 2 pakiety: *UI* oraz *Logic*. Pierwszy z pakietów odpowiada za komunikację z użytkownikiem (wczytanie parametrów łamigłówki oraz prezentację rozwiązania). Pakiet *Logic* odpowiada za znalezienie rozwiązania łamigłówki.

Komunikacja z użytkownikiem

Na początku działania programu wczytywane są parametry łamigłówki z pliku. Za wczytanie oraz przetworzenie danych z pliku odpowiedzialna jest funkcja *loadPuzzle* z modułu *UI.PuzzleLoader*.

Za prezentację wyniku odpowiada funkcja *prezentResult* z modułu *UI.ResultPrezenter*. Funkcja ta najpierw wypisuje w konsoli rozwiązanie, a następnie zapisuje wynik do wskazanego przez użytkownika pliku. Za wypisanie wyniku na ekranie odpowiada funkcja *printSolution* z modułu *UI.ResultPrinter*.

Poszukiwanie rozwiązania

Poszukiwanie rozwiązania przypomina przeszukiwanie wgłąb drzewa, w którym węzły stanowią rozwiązania (częściowe lub całkowite, niekoniecznie poprawne). Rozwiązaniem całkowitym jest takie rozwiązanie, w którym każdy dom ma przydzielony swój zbiornik. Przy przyjęciu, że „przestrzeń" rozwiązań ma postać drzewa, rozwiązaniami całkowitymi są „liście" drzewa.

Funkcją odpowiedzialną za znalezienie rozwiązania jest funkcja *findSolution'* z modułu *Logic.SolutionFinder*. Ma ona postać rekurencyjną, gdzie warunkiem końcowym rekurencji jest znalezienie rozwiązania całkowitego lub otrzymanie rozwiązania błędnego (częściowego lub całkowitego). Funkcja *findSolution'* jako jeden z parametrów przyjmuje częściowe rozwiązanie (lista przydziałów zbiornik→dom). Dlatego moduł *Logic.SolutionFinder* udostępnia funkcję *findSolution* wywołującą funkcję *findSolution'* z pustą listą przydziałów.

Funkcja *findSolution'* zawszae na początku sprawdza czy otrzymane rozwiązanie jest poprawne i jeśli wszystkie domy mają przydzielone zbiorniki, zwraca otrzymane rozwiązanie. Jeśli otrzymane rozwiązanie jest błędne, zwracana jest lista pusta, co oznacza porażkę w znaleziu rozwiązania.

Jeśli funkcja *findSolution'* otrzyma niepustą listę domów, wybiera jeden z domow, i wywołuje dla niego funkcję *movesIter*, której wynik jest zwracany.

Funkcja *movesIter* ma za zadanie znaleźć zbiornik dla zadanego domu iterując po możliwych pozycjach zbiornika względem domu, a następnie uruchomić *findSolution'* dla pozostałych domów. Jednym z argumentow otrzymywanych przez *movesIter* jest lista możliwych pozycji zbiornika (przy wywołaniu najwyższego poziomu jest to lista wszystkich możliwych ruchów, niekoniecznie poprawnych), która jest zmniejszana wraz z iterowaniem po kolejnych możliwych pozycjach zbiornika. Jeśli wywoływana w *movesIter* funkcja *findSolution'* zwróci rozwiązanie poprawne, wówczas jest ono zwracane. W przeciwnym wypadku badane są kolejne dostępne ruchy lub zwracana jest lista pusta (jeśli nie ma więcej możliwych ruchów).

Za sprawdzanie poprawności rozwiązania odpowiada funkcja *hasSolutionErrors* z modułu *Logic.Constraints*. Sprawdza ona czy nie zostały złamane ograniczenia na liczbę zbiorników w wierszu/kolumnie, czy nowododany zbiornik nie koliduje z innymi obiektami i czy nie jest poza planszą. Dodatkowo w module *Logic.Constraints* udostępniane są funkcje *updateRowConstr* oraz *updateColumnConstr* aktualizujące ograniczenia na liczbę zbiorników w wierszach i kolumnach po wykonaniu wybranego ruchu.