Sofia Krutko, Sylwia Kulesza, projekt PZ.P.2 - Country road

Opis zadania

Napisać aplikację rozwiązującą łamigłówkę country road na kwadratowej planszy o dowolnym rozmiarze. Aplikacja powinna umożliwiać wczytywanie instancji łamigłówki z pliku, rozwiązywanie jej przez człowieka, jak i automatyczne generowanie rozwiązania bazujące na algorytmie A*.
Reguły gry Country Road: build one big looping road that goes through the squares of the map. Every country (marked by thicker lines) has to be visited exactly once, not more. All of the borders between countries have to have roads on at least one side: two squares with a piece of border between them cannot both stay empty.

Opis realizacji

Aby uruchomić grę należy podać: java Main *nazwa_pliku.txt* 1
albo java Main *nazwa_pliku.txt* 2
Gra zawiera dwa tryby: tryb 1 - aplikacja uruchamia się pokazując automatyczne rozwiązywanie łamigłówki.
Tryb 2 - aplikacja nie pokazuje od razu rozwiązania. Można od razu rysować albo kliknąć przycisk reveal który pokaże rozwiązanie automatyczne.

Są cztery przygotowane pliki: pierwsza_plansza.txt, druga_plansza.txt, trzecia_plansza.txt, czwarta_plansza.txt

GameBoard:
Zawiera dwuwymiarową tablicę squares o rozmiarze nxn, gdzie n to rozmiar planszy.

Liczba n jest pobierana z pliku z instancją łamigłówki. Jest to pierwsza cyfra zapisana w pliku. Dalej są macierze oznaczające kolejno - pionowe kreski na planszy (1 oznacza grubą granice państwa, 0 oznacza cienką linię na planszy), poziome linie na planszy (1 i 0 tak samo jak w pionowych) i na końcu oznaczenie państw cyframi, zaczynając od 0.

Czyli: są trzy rodzaje granic:

```
BORDER(-1),
INTERNAL(0),
EXTERNAL(1);
```

Border oznacza zewnętrzną granicę planszy. External - **gruba** granica pomiędzy państwami, Internal - cienka linia na planszy.

Square:
Oznacza pojedynczy kwadracik na planszy. Ma swoją granicę górną, dolną,lewą i prawą. Ma też indeks państwa do którego należy.

A*:
W klasie AStar w funkcji solve() jest wybierany początkowy punkt (Square) działania algorytmu (ten punkt również jest punktem końcowym). O ile rozwiązaniem jest drogą zamkniętą, pierwszy punkt nie ma wpływu na działanie algorytmu, jeśli wybrany kwadrat należy do rozwiązania, zostanie ono odnalezione.
Najpierw wybierany jest kwadrat (0,0), jeśli rozwiązanie nie było znalezione, to następny kwadrat (0,1) i t.d. Jest wywoływana funkcja AStar, której są przekazywane współrzędne początkowego punktu działania.

W funkcji AStar jest tworzony wektor openSquares, który odpowiada za przechowywanie całej przestrzeni stanów, do których dotarł algorytm, ale które jeszcze nie zostały rozwinięte. Reprezentują te stany obiekty typu Cell - pomocnicze kopii kwadratów planszy. Głownymi atrybutami pomocniczych kopii są parent, costC i costCh.
Parent - to obiekt typu Cell, z którego droga przychodzi na dany kwadrat. Warto zauważyć, że w procesie poszukiwania drogi zamkniętej, jest możliwe trafienie do danego kwadratu na każdym etapie poszukiwania, dlatego istnieje potrzeba dla każdego następnego trafienia do niego tworzyć oddzielny obiekt pomocniczy - czyli oddzielny stan.
costC - koszt dotarcia do danego kwadratu z punktu początkowego.
costCh - koszt dotarcia do danego kwadratu + dolne oszacowanie dotarcia do końca.
Schemat działania funkcji AStar:

```
function AStar(start)
	openSquares := set containing the node (start.right_neighbor)
	solution := { start, start.right_neighbor}

	while openset is not empty and solution is incorrect
		current := the node in openset having the lowest costCh value
		remove current from openset
		if current.left_neighbor can be added to openSquares
			costCh = current.costC + 1 + h(current.left_neighbor)
			add to openSquares
		if current.bottom_neighbor can be added to openSquares
			costCh = current.costC + 1 + h(current.bottom_neighbor)
			add to openSquares
		if current.right_neighbor can be added to openSquares
			costCh = current.costC + 1 + h(current.right_neighbor)
			add to openSquares
		if current.upper_neighbor can be added to openSquares
			costCh = current.costC + 1 + h(current.upper_neighbor)
			add to openSquares
		solution = reverse_path( current, start )

function reverse_path(current, start)
	if current = start
		path = { start }
		return path
```

```
path = reverse_path(current.parent, start)
add to path current
return path
```

Sąsiad danego kwadratu może być dodany na listę openSquares jeśli on:

A. nie należy do ścieżki, która prowadzi do danego kwadratu;
B. nie należy do kraju, który już został odwiedzony.

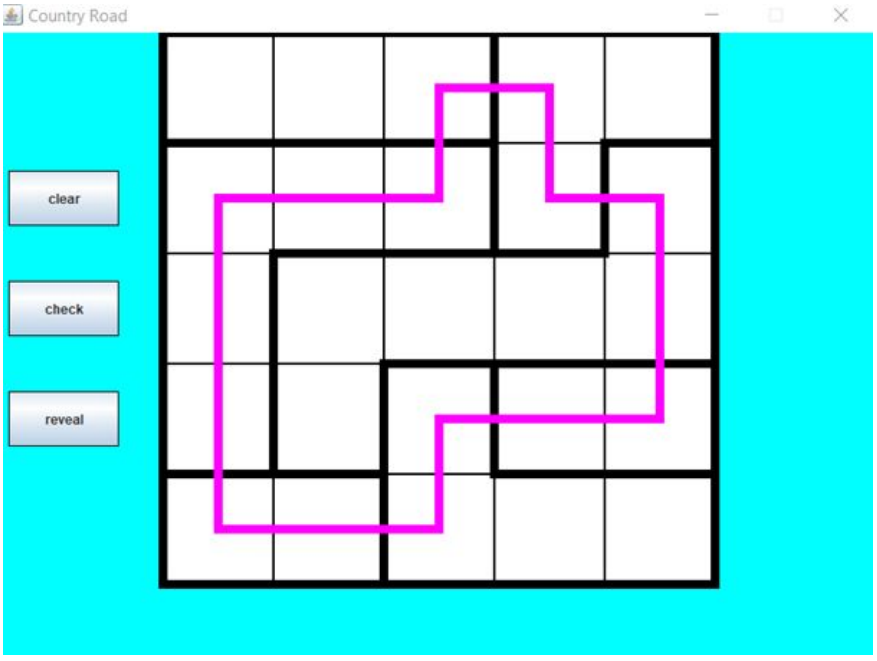
Natomiast czy ten sąsiad już był odwiedzony w ramach innego rozwiązania nie jest wazne.
Te warunki są sprawdzane w funkcji checkNextCell.

Klasa View odpowiada za widok (rysowanie planszy, rozwiązania, możliwość rysowania drogi przez użytkownika, zdefiniowanie przycisków.)
Są trzy przyciski:
clear, reveal, oraz check. Clear ma wyczyścić drogę narysowaną na planszy. Reveal ma pokazać rozwiązanie generowane automatycznie a check - sprawdzić rozwiązanie wprowadzone przez użytkownika.

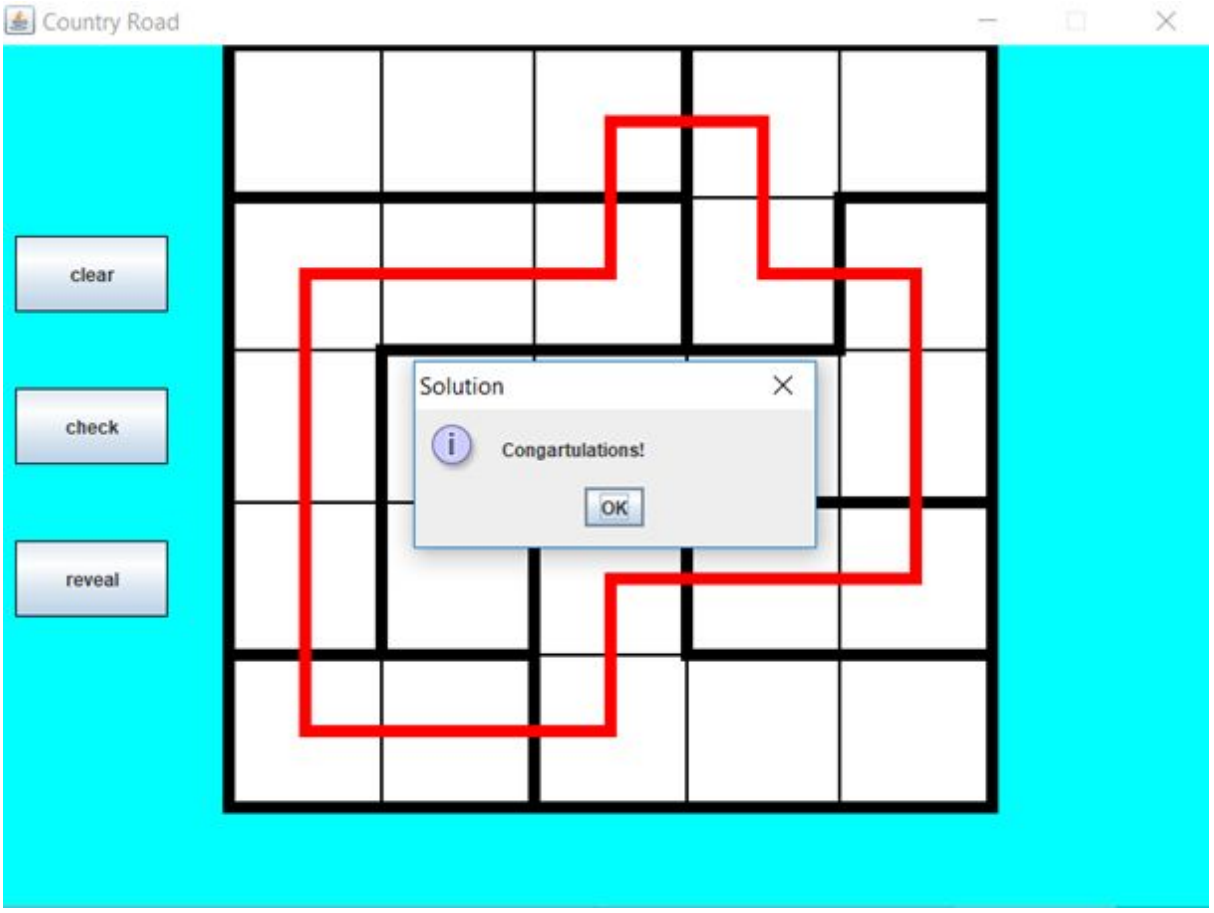
Wyniki działania

Poniżej zamieszczone są screeny pokazujące działanie programu:

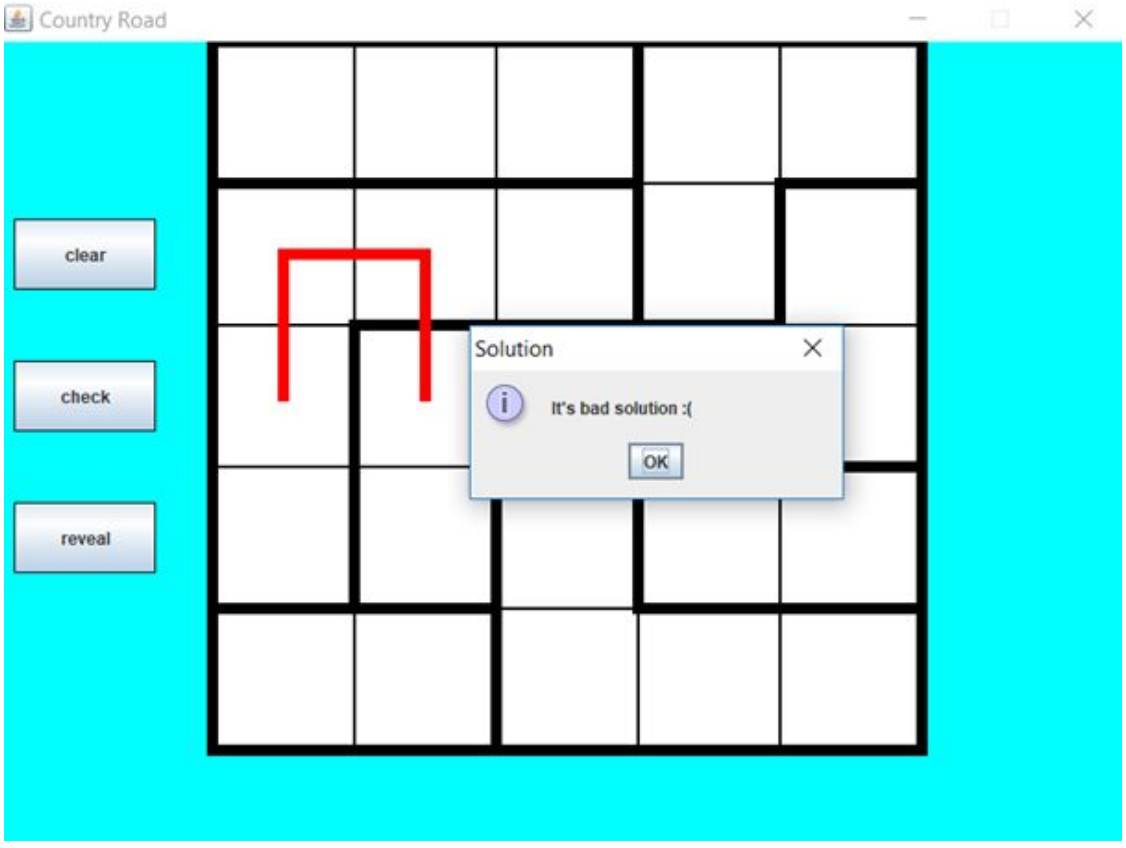
pierwsza_plansza.txt



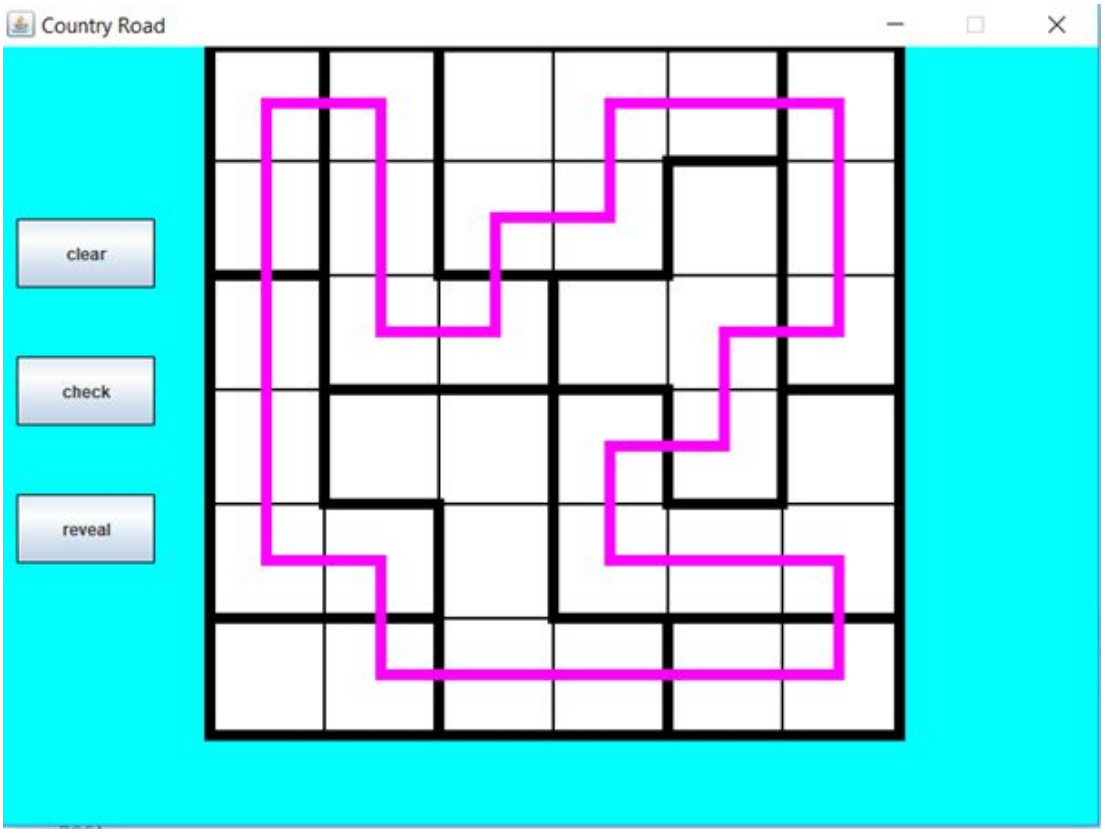
Sama łamigłówka ale rozwiązana przez użytkownika:



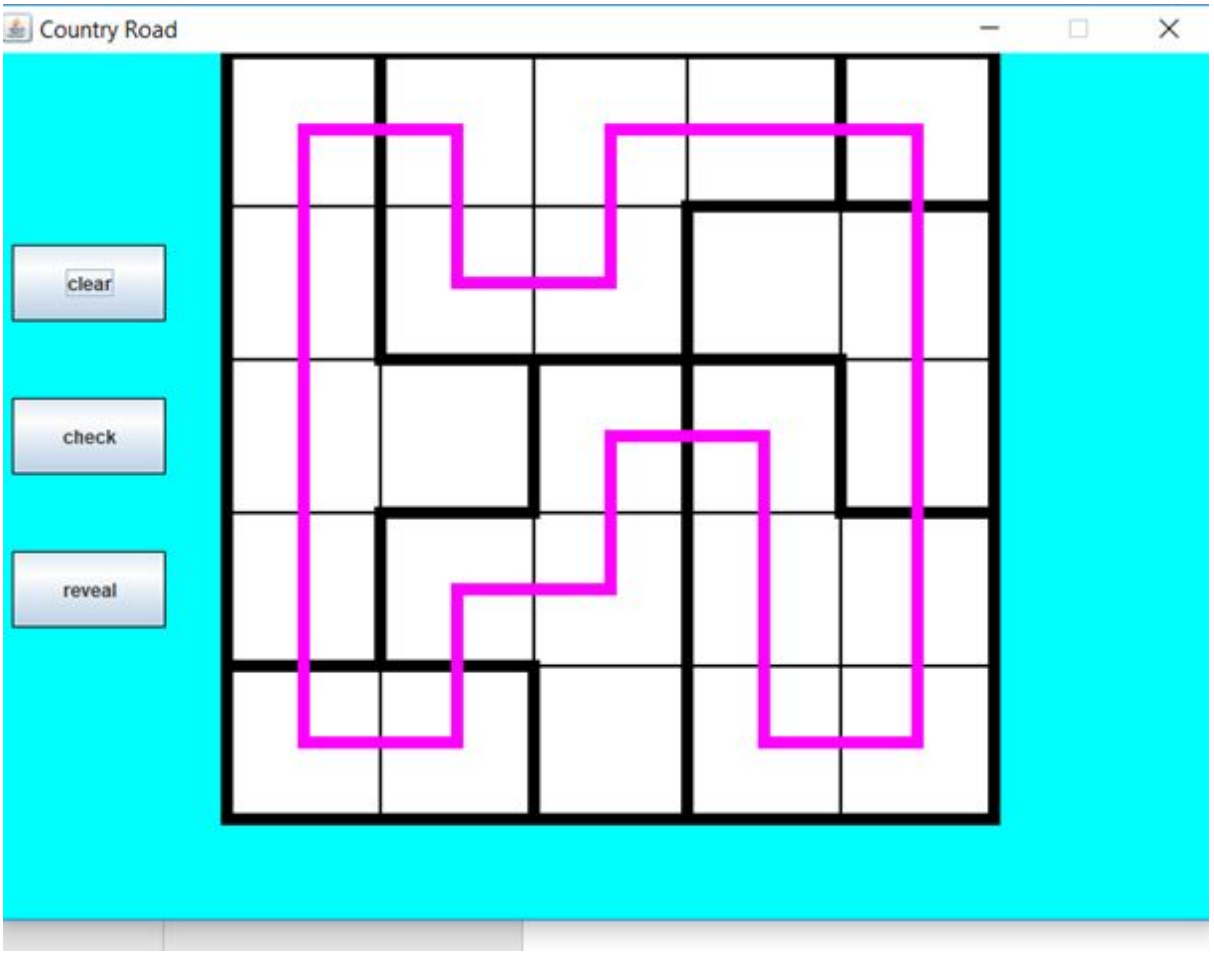
Przykład kiedy rozwiązanie użytkownika jest złe:



druga_plansza.txt



trzecia_plansza.txt



czwarta_plansza.txt