Санкт-Петербургский политехнический университет Петра Великого

Институт компьютерных наук и технологий

Высшая школа интеллектуальных систем и суперкомпьютерных технологий

**КУРСОВОЙ ПРОЕКТ**

Разработка программы для поиска кратчайшего пути в лабиринте

по дисциплине «Алгоритмы и структуры данных»

Выполнил

студент гр. 3530901/90003

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Мощенских М.А.

(подпись)

Руководитель

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Ахин М.Х.

(подпись)

«\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2020 г.

Санкт-Петербург   
2020

Санкт-Петербургский политехнический университет Петра Великого

**ЗАДАНИЕ**

**НА ВЫПолнение курсовоГО ПРОЕКТА**

студенту группы 3530901/90003 Мощенских Максиму Александровичу

***1. Тема проекта (работы)***: Разработка программы для поиска кратчайшего пути в лабиринте

***2. Срок сдачи законченной работы***: 30 декабря

***3. Исходные данные к работе***: варианты курсового проекта, вариант 27

***4. Содержание пояснительной записки***: техническое задание, метод решения, описание проведённых тестов, описание работы программы, заключение, список использованных источников.

***Дата получения задания***: «2». сентября 2020 г.

Руководитель \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Ахин М.Х.

(подпись)

Задание принял к исполнению \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Мощенских М.А.

(подпись)

\_\_\_\_\_\_\_\_\_\_\_\_

(дата)

**ТЕХНИЧЕСКОЕ ЗАДАНИЕ**

**Исходные данные:** лабиринт задан матрицей NxM со значениями 0 или 1 (0 – проход, 1 – стенка). Найти самый короткий путь между заданными "точками" Start (S) и Finish (F). Графическое представление задачи приведено на рис. 1.

|  |
| --- |
|  |
| Рис. 1. Вид лабиринта |

**Требования к программе:** готовая программа должна представлять собой приложение с графическим интерфейсом пользователя (GUI) на языке Java.

В главном окне программы должен быть изображён лабиринт, состоящий из отдельных клеток; нажатием на клетку можно поменять её тип (стенка/проход).

Должны присутствовать кнопки для задания начальной (S) и конечной (F) клеток лабиринта и кнопка для поиска кратчайшего пути между заданными начальной и конечной клетками (Start), при нажатии на которую подсвечиваются клетки, составляющие этот путь.

Если начальная или конечная точки не заданы, нельзя решать лабиринт (блокируется кнопка Start).

Также должны присутствовать кнопка для генерации случайного лабиринта, кнопка сброса и выпадающие списки для изменения размеров лабиринта по вертикали и горизонтали.

Ссылка на репозиторий: https://github.com/mmoschenskikh/shortest-maze-path

**МЕТОД РЕШЕНИЯ**

Для разработки были использованы следующие средства и технологии:

* язык программирования Java для написания кода программы;
* библиотека JavaFX для создания графического интерфейса (GUI);
* программа Scene Builder для создания файлов разметки .fxml;
* библиотека JUnit для создания автоматических тестов;
* система сборки Maven для управления зависимостями и создания .jar‑файлов;
* система контроля версий git;
* среда разработки IntelliJ IDEA Community Edition;
* шаблон MVC.

Построение приложения по шаблону MVC (Model-View-Controller) позволяет разделить исходный код на три компонента: данные приложения (модель), пользовательский интерфейс (представление) и логика, связывающая представление и модель (контроллер). Приложение содержит четыре пакета:

1. ru.spbstu.shortestmazepath

Содержит в себе остальные пакеты, а также классы для запуска приложения:

* Main, имеющий только метод main(String[]) и служащий для упрощения создания .jar-файлов системой Maven. Вызов метода запускает приложение с помощью класса MazeApp.
* MazeApp, имеет методы main(String[]) для запуска приложения и start(Stage) для инициализации параметров окна приложения.

1. ru.spbstu.shortestmazepath.util

Содержит единственный класс StringsSupplier, реализующий паттерн «одиночка» (Singleton) и предназначенный для упрощения доступа к строковым ресурсам из любого места программы. Единственный метод класса, getStrings(), возвращает объект типа ResourceBundle, содержащий пары ключ-значение для строковых ресурсов программы.

1. ru.spbstu.shortestmazepath.model

Содержит классы, описывающие логику программы:

* Класс Cell представляет клетку лабиринта, имеющую координаты x и y, и тип, представленный вложенным классом-перечислением Cell.Type. Класс Cell.Type имеет лишь метод toString(), возвращающий строковое представление типа. Класс Cell содержит следующие методы:
  + Cell(int, int, Cell.Type) — конструктор.
  + Переопределённые методы класса Object — equals(Object), hashCode() и toString().
  + distanceTo(Cell) принимает в качестве аргумента объект класса Cell и возвращает расстояние между двумя точками. Расстояние вычисляется по теореме Пифагора (евклидово расстояние).
* Класс Maze представляет лабиринт. Содержит следующие методы:
  + Maze(int, int, Cell, Cell, Cell[][]) — конструктор.
  + getHeight() и getWidth() возвращают высоту и ширину лабиринта соответственно.
  + getStartCell() и getEndCell() возвращают начальную и конечную точку в заданном лабиринте соответственно.
  + getMazeGrid() возвращает двумерный массив объектов Cell, представляющий лабиринт.
  + Переопределённые методы класса Object — equals(Object), hashCode() и toString().
  + getNeighbours(Cell) возвращает множество соседних с переданной в качестве аргумента клеток. Клетка является соседом, если не представляет собой стену.
  + getHeuristics(Cell) возвращает расстояние от переданной в качестве аргумента клетки до конечной клетки.
  + random() возвращает созданный случайно лабиринт. Не гарантируется, что в нём будет проход от начальной клетки к конечной.
  + solve() решает лабиринт, используя **алгоритм A\*** и возвращает список клеток, которые содержит кратчайший путь между заданными начальной и конечной точками.
* Класс MazeManager предоставляет возможности для работы с файлами лабиринтов на диске. Содержит два метода:
  + save(Maze, File) записывает переданный в качестве аргумента лабиринт в указанный файл. Для записи в файл используется строковое представление типа каждой клетки (Cell.Type.toString()).
  + load(File) загружает лабиринт из указанного файла и возвращает новый объект лабиринта.

1. ru.spbstu.shortestmazepath.controller

Содержит единственный класс MazeController, связывающий логику и графический интерфейс программы. Содержит вложенный класс-перечисление MazeController.Type, метод identify(ImageView) по изображению клетки определяет её тип. Класс служит для упрощения перевода графического представления лабиринта во внутреннее представление в классе Maze. Класс MazeController содержит следующие методы:

* initialize(URL, ResourceBundle) — предназначен для задания начального состояния окна и установки слушателей для необходимых компонентов окна.
* initializeMaze() подготавливает сетку GridPane и заполняет её для представления пустого лабиринта размера 7x7.
* setGridPaneConstraints(int) задаёт размеры ячейки в GridPane, делая её квадратной.
* getSizeOptionsList() возвращает список с возможными размерами лабиринта (для инициализации выпадающих списков).
* addMazeRow() добавляет ряд в лабиринт и заполняет его клетками прохода.
* addMazeColumn() добавляет столбец в лабиринт и заполняет его клетками прохода.
* checkStartEndSet() управляет доступностью кнопки “Solve”, включая её, если начальная и конечная клетки заданы и отключая в противном случае.
* checkNotOnField(Node) возвращает логическое значение, показывающее, видна ли клетка на экране в данный момент.
* setMazeHeight(int) изменяет высоту лабиринта в соответствии с переданным значением.
* setMazeWidth(int) изменяет ширину лабиринта в соответствии с переданным значением.
* prepareMazeCell() возвращает ImageView клетки лабиринта с заданной логикой для нажатия.
* onStartSet() позволяет задать начальную точку.
* onEndSet() позволяет задать конечную точку.
* onRandom() генерирует и отображает случайный лабиринт.
* resetMaze() приводит лабиринт в первоначальное состояние (пустой лабиринт размером 7x7).
* onReset() обрабатывает нажатие на кнопку “Reset”, во избежание потери данных пользователь должен подтвердить своё намерение.
* onExit() обрабатывает нажатие на попытку выхода из программы, во избежание потери данных пользователь должен подтвердить своё намерение.
* showResetConfirmation(String, Consumer<?>) показывает диалоговое окно для подтверждения сброса лабиринта. В случае положительного ответа выполняется переданное в качестве параметра действие.
* showErrorMessage(String) показывает пользователю сообщение об ошибке.
* onSolve() решает лабиринт и отображает решение.
* hideSolution() скрывает решение, если клетки решения подсвечены.
* setSolutionOpacity(List<Cell>, double) меняет прозрачность для набора клеток, метод предназначен для подсвечивания клеток решения.
* prepareFileChooser() подготавливает окно выбора файла.
* onLoad() загружает выбранный пользователем файл и отображает содержащийся в нём лабиринт.
* onSave() сохраняет текущий лабиринт в файл.
* loadMaze(Maze) отображает лабиринт, представленный классом Maze, на экране.
* mazeViewToModel() возвращает объект Maze, представляющий отображаемый на экране лабиринт.

**ТЕСТИРОВАНИЕ ПРИЛОЖЕНИЯ**

Для проверки правильности работы отдельных компонентов приложения были созданы автоматические тесты. Данные тесты охватывают все классы пакета ru.spbstu.shortestmazepath.model, а также единственный класс пакета ru.spbstu.shortestmazepath.util.

Кроме автоматических тестов вручную была протестирована правильность работы GUI, например, сохранение и загрузка файлов, взаимодействие с элементами графического интерфейса.

**РАБОТА ПРОГРАММЫ**

Работа программы продемонстрирована на рис. 1 – 7. В названиях рисунка описана ситуация, возникающая в ходе работы программы.

Выпадающие списки Height и Width позволяют задавать высоту и ширину лабиринта соответственно. Кнопка Randomize отображает случайно сгенерированный лабиринт (рис. 2). Кнопки Set Start Point и Set End Point позволяют задать начальную и конечную точки соответственно. Кнопка Reset Maze сбрасывает лабиринт к начальному состоянию (рис. 1). Кнопка Solve находит кратчайший путь в лабиринте и отображает его, если он есть, на экране (рис. 4, 5). Меню File позволяет загружать и сохранять лабиринты (рис. 3, 6, 7). Информацию о текущем состоянии программы можно получить из строки состояния в левом нижнем углу окна.

|  |
| --- |
|  |
| Рис. 1. Начальный экран |

|  |
| --- |
|  |
| Рис. 2. Сгенерированный случайно лабиринт |

|  |
| --- |
|  |
| Рис. 3. Загруженный из файла лабиринт |

|  |
| --- |
|  |
| Рис. 4. Нахождение кратчайшего пути в лабиринте |

|  |
| --- |
|  |
| Рис. 5. Попытка решить лабиринт без прохода между заданными клетками |

|  |
| --- |
|  |
| Рис. 6. Попытка загрузки файла с некорректным содержанием |

|  |
| --- |
|  |
| Рис. 7. Попытка загрузки файла с некорректным содержанием |

**ЗАКЛЮЧЕНИЕ**

В ходе работы над курсовым проектом было создано приложение с графическим интерфейсом, позволяющее находить кратчайший путь в лабиринте. Библиотека JavaFX имеет большое количество готовых элементов GUI, что даёт возможность разрабатывать достаточно сложные интерфейсы. Шаблон MVC позволил разделить код приложения на части, что сделало удобной работу с каждой из них и облегчило разработку.

**СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

1. https://neerc.ifmo.ru/wiki/index.php?title=%D0%90%D0%BB%D0%B3%D0%BE%D1%80%D0%B8%D1%82%D0%BC\_A\* — описание алгоритма A\*.
2. <https://docs.oracle.com/javase/8/javafx/api/> — javadoc по JavaFX.
3. Шилдт Г. Java. Полное руководство, 10-е изд. : Пер. с англ. – СПб: «Альфа‑книга», 2018 – 1488 с.
4. Vos, Gao. Pro JavaFx 8. – Apress, 2014 – 604 с.