Министерство Цифрового Развития, Связи и Массовых Коммуникаций Российской Федерации Ордена Трудового Красного Знамени федеральное государственное бюджетное образовательное учреждение высшего образования   
“Московский технический университет связи и информатики”

Кафедра «Информационные технологии»

Лабораторная работа №3:   
«Алгоритм A\* («A star»)**»**

Выполнил: студент гр. БПЗ1902

Оглоблин Д.И.

Проверил: Харрасов К.Р.

Москва 2021 г.

Цель работы:

В данной лабораторной работе необходимо изучить программу поиска пути, и исключить возникновение проблем при создании препятствий и последующим поиском путей вокруг них.

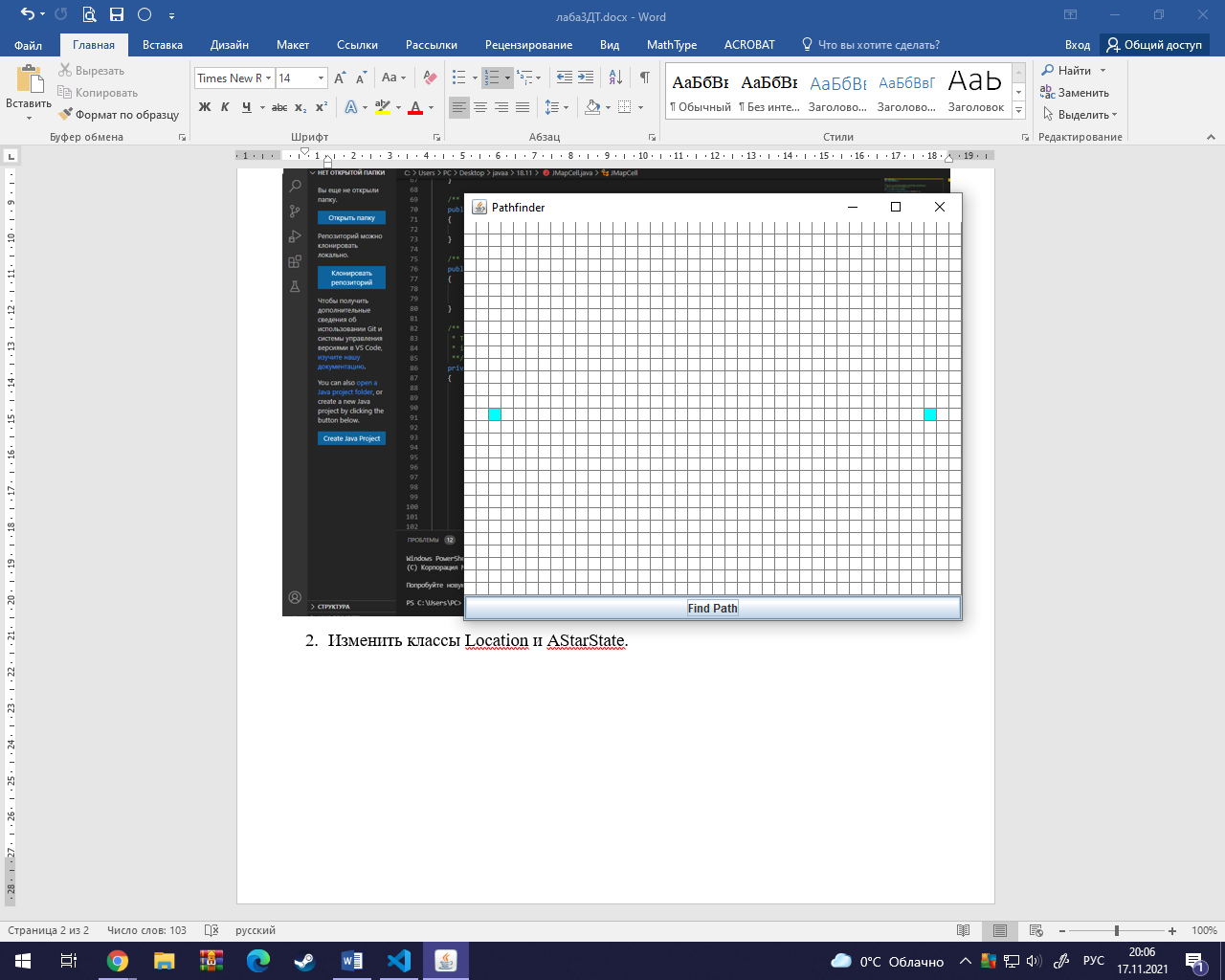
Задание:

1. Скачать исходные файлы для данной лабораторной работы.
2. Изменить классы Location и AStarState.

Выполнение работы:

1. Скачать исходные файлы для данной лабораторной работы.

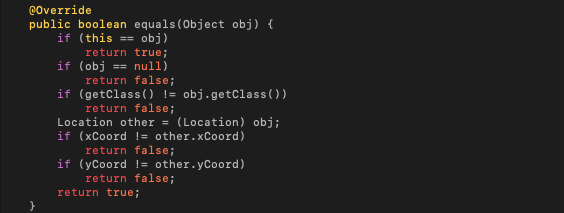
Приложение успешно компилируется в том виде, какое оно есть, но функция поиска пути не работает.



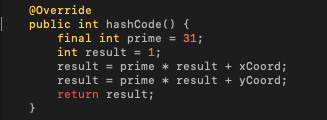
1. Изменить классы Location и AStarState.

Location:

Для начала необходимо подготовить класс Location для совместного использования с классами коллекции Java. Мы будем использовать контейнеры для хеширования для выполнения данного задания, для этого необходимо:   
• Обеспечить реализацию метода equals ().



• Обеспечить реализацию метода hashcode().



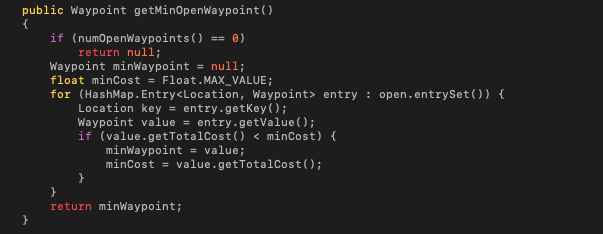
Код Location:

Теперь мы можем использовать класс Location в качестве ключевого типа в контейнерах хеширования, таких как HashSet и HashMap.

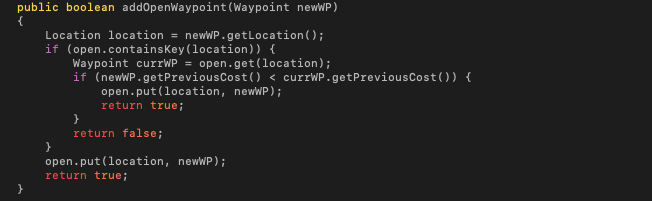
AStarState:

Добавим два (нестатических) поля в класс AStarState с таким типом, одно для "открытых вершин" и другой для "закрытых вершин".   
Кроме того, инициализируем каждое из этих полей для ссылки на новую пустую коллекцию. После создания и инициализации полей, мы реализуем следующие методы в классе AStarState:   
1) public int numOpenWaypoints() Этот метод возвращает количество точек в наборе открытых вершин.

  
2) public Waypoint getMinOpenWaypoint() Эта функция должна проверить все вершины в наборе открытых вершин, и после этого она должна вернуть ссылку на вершину с наименьшей общей стоимостью. Если в "открытом" наборе нет вершин, функция возвращает NULL. Не удаляйте вершину из набора после того, как вы вернули ее; просто верните ссылку на точку с наименьшей общей стоимостью.

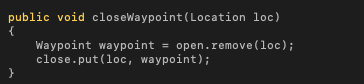
  
3) public boolean addOpenWaypoint(Waypoint newWP) Это самый сложный метод в классе состояний А\*. Данный метод усложняет то, что он должен добавлять указанную вершину только в том случае, если существующая вершина хуже новой. Вот что должен делать этот метод:

• Если в наборе «открытых вершин» в настоящее время нет вершины для данного местоположения, то необходимо просто добавить новую вершину.  
 • Если в наборе «открытых вершин» уже есть вершина для этой локации, добавьте новую вершину только в том случае, если стоимость пути до новой вершины меньше стоимости пути до текущей. (Убедитесь, что используете не общую стоимость.) Другими словами, если путь через новую вершину короче, чем путь через текущую вершину, замените текущую вершину на новую



4) public boolean isLocationClosed(Location loc) Эта функция должна возвращать значение true, если указанное местоположение встречается в наборе закрытых вершин, и false в противном случае. Так как закрытые вершины хранятся в хэш-карте с расположениями в качестве ключевых значений, данный метод достаточно просто в реализации.

  
5) public void closeWaypoint(Location loc) Эта функция перемещает вершину из набора «открытых вершин» в набор «закрытых вершин». Так как вершины обозначены местоположением, метод принимает местоположение вершины. Процесс должен быть простым:   
• Удалите вершину, соответствующую указанному местоположению из набора «открытых вершин».   
• Добавьте вершину, которую вы удалили, в набор закрытых вершин. Ключом должно являться местоположение точки.



Компиляция и тестирование:

При компиляции не возникает проблем при создании препятствий и последующим поиском путей вокруг них.

