**Федеральное государственное общеобразовательное учреждение высшего образования**

**Московский государственный технический университет**

**им. Н. Э. Баумана**

Факультет "Информатика и системы управления"

Кафедра "Информационные системы и телекоммуникации"

**Отчет по лабораторной работе №3**

**"Алгоритм A"**

Студент: Задубин А.А.

Группа: ИУ3-42Б

Преподаватель: Руденкова Ю.С.

**Задание:**

Нам требовалось внести изменения в текст двух классов: Location и AStarState. Все остальное это код оболочки Swing который позволяет нам редактировать карту и отображает маршрут рассчитанный алгоритмом. Позиции (класс Location)

Следовало начать с подготовки класса Location для использования внутри классов коллекций Java. Так как в этой работе использовались хэш контейнеры, для этого требовалось:

• Добавить реализацию метода equals().

• Добавить реализацию метода hashCode().

Добавили эти методы к классу Location, следуя шаблону внутри класса. После компиляции, класс Location можно использовать как тип ключа в хэш контейнерах, таких как HashSet и HashMap.

Состояние A\* (Класс AStarState)

Как только класс Location бул готов для использования в качестве ключа, завершили реализацию класса AStarState. Этот класс содержит множества открытых и закрытых точек маршрута, и обеспечивает базовый функционал для реализации алгоритма A\*. Состояние A\* включает две коллекции точек маршрута, одну для открытых точек и другую для закрытых точек. Для упрощения алгоритма, точки маршрута хранятся в ассоциативном хэш списке , в котором ключами являются позиции, а значениями точки маршрута. Мы приходим к такому типу: HashMap. Добавили два (не статических) поля этого типа к классу AStarState, одно для открытых точек маршрута, другое для закрытых. Не забыли проинициализировать оба поля так, чтобы они указывали на новые пустые коллекции. После того как поля добавлены и правильно проинициализированы, следовало добавить следующие методы к классу AStarState: *public int numOpenWaypoints()*. Этот метод должен был просто возвращать число элементов в коллекции открытых точек маршрута.

*public Waypoint getMinOpenWaypoint()* Эта функция должна былв перебирать все элементы в коллекции открытых точек маршрута и возвращать ссылку на точку с наименьшим значением стоимости маршрута. Если в коллекции нет элементов, метод должен был возвращать null. Не удаляем точку из коллекции, возвращая ее; возвращаем только ссылку на точку с минимальной стоимостью маршрута.

*public boolean addOpenWaypoint(Waypoint newWP)* Это самый сложный метод класса состояния A\*, но его реализация довольно проста. Сложнее других его делает то, что в нем надо было добавить новую точку маршрута, если существующая точка маршрута в позиции “хуже”, чем новая. Вот что должен был делать метод:

• Если в позиции нет открытой точки маршрута, просто добавили новую точку.

• Если в коллекции открытых точек уже есть точка маршрута в такой позиции, добавили новую точку.

Здесь, как видно, требовалось найти точку маршрута в коллекции открытых точек и, если она там присутствует, возможно, заменить ее. К счастью это очень просто делается с помощью метода HashMap.put(), который заменяет старую ассоциацию ключ-значение на новую. Метод должен был возвращать true, если в коллекцию попадает новая точка маршрута, и false, если этого не происходит.

*public boolean isLocationClosed(Location loc)* Эта функция возвращает true если указанная позиция имеется в коллекции закрытых точек маршрута, иначе false. Так как зарытые точки хранятся в ассоциативном хэш списке в котором позиция это ключ , сделать такой метод очень просто.

*public void closeWaypoint(Location loc)* Эта функция перемещает точку маршрута из коллекции открытых точек в коллекцию закрытых точек. Так как точку можно найти по ее позиции, методу передается позиция точки. Процесс перемещения точки выглядит так:

• Удалили точку в указанной позиции из коллекции открытых точек.

• Добавили удаленную точку в коллекцию закрытых точек. Конечно с ключом будет позиция точки.

**Код программы:**

*Класс AStarApp.java*

**public** **class** **AStarState** {

**private** Map2D map;

**private** HashMap<Location, Waypoint> open = **new** HashMap<Location, Waypoint>();

**private** HashMap<Location, Waypoint> closed = **new** HashMap<Location, Waypoint>();

**public** **AStarState**(Map2D map) {

**if** (map == **null**)

**throw** **new** **NullPointerException**("map cannot be null");

**this**.map = map;

}

**public** Map2D **getMap**() {}

**public** Waypoint **getMinOpenWaypoint**() { }

**public** **boolean** **addOpenWaypoint**(Waypoint newWP) {}

**public** **int** **numOpenWaypoints**() {}

**public** **void** **closeWaypoint**(Location loc) {}

**public** **boolean** **isLocationClosed**(Location loc) {}

}

*Класс Location.java*

**public** **class** **Location** {

**public** **int** xCoord;

**public** **int** yCoord;

**public** **Location**(**int** x, **int** y) {

xCoord = x;

yCoord = y;

}

**public** **Location**() {

**this**(**0**, **0**);

}

**@Override**

**public** **boolean** **equals**(Object obj) {

Location loc = (Location) obj;

**return** loc.xCoord == xCoord && loc.yCoord == yCoord;

}

**public** **boolean** **equals2**(Location obj) {

**return** obj.xCoord == xCoord && obj.yCoord == yCoord;

}

**@Override**

**public** **int** **hashCode**() {

**final** **int** prime = **32**;

**int** result = **1**;

result = prime \* result + xCoord;

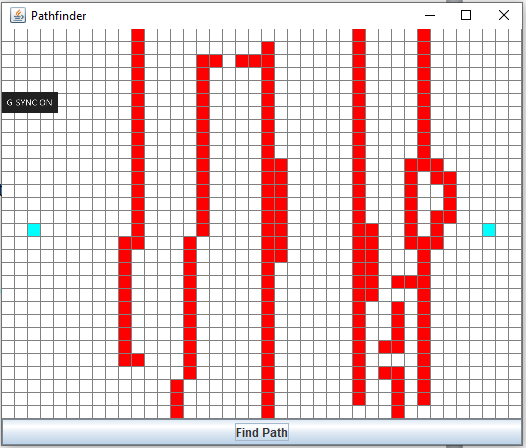
result = prime \* result + yCoord;

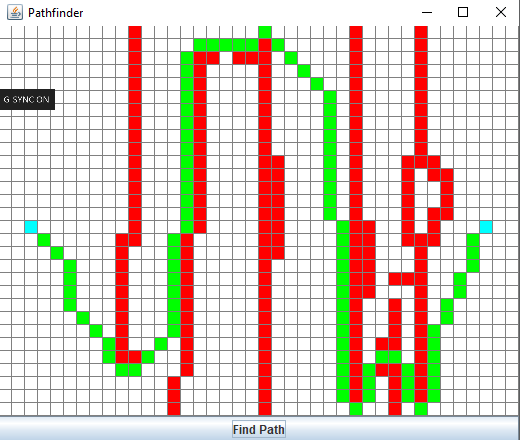
**return** result;

}

}

**Результат работы программы:**





**Вывод**

В ходе данной лабораторной работы мы научились работать с коллекциями классов Java, изучили работу алгоритма А\* и доработали код программы, которая ищет кратчайший путь на карте.