**МИНИСТЕРСТВО ЦИФРОВОГО РАЗВИТИЯ СВЯЗИ И МАССОВЫХ КОММУНИКАЦИЙ**

**Ордена Трудового Красного Знамени**

**Федеральное государственное бюджетное образовательное учреждение высшего образования**

**«Московский технический университет связи и информатики»**

Кафедра «Математическая кибернетика и информационные технологии»

**Отчет по лабораторной работе № 3**

по дисциплине «Информационные технологии и программирование»

на тему:

Алгоритм A\* («A star»)

Выполнила:

студентка группы БВТ2104

Степанова Дарина Викторовна

Проверила:

Мосева Марина Сергеевна

Москва, 2022

**Цель работы**: необходимо использовать хранилище java.util.HashMap для каждого из наборов алгоритма А\* с объектами Location в качестве ключей, и объектами Waypoint в качестве значений.

**Задания:**

1. Обеспечить реализацию метода equals ().
2. Обеспечить реализацию метода hashcode().
3. После создания и инициализации полей реализовать 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 нужно удалять вершину из набора после того, как вернули ее; просто вернуть 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. (Убедиться, что используется не общая стоимость.) Другими словами, если путь через новую вершину короче, чем путь через текущую вершину, заменить текущую вершину на новую. В таком случае потребуется извлечь существующую вершину из «открытого набора», если таковая имеется. Данный шаг довольно прост - заменить предыдущую точку на новую, используя метод HashMap.put(), который заменит старое значение на новое. Пусть данный метод вернет значение true, если новая вершина была успешно добавлена в набор, и false в противном случае.

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 вершин

1. Скомпилировать и запустить программу также, как и всегда:

javac \*.java

java AStarAp

**Код программы:**

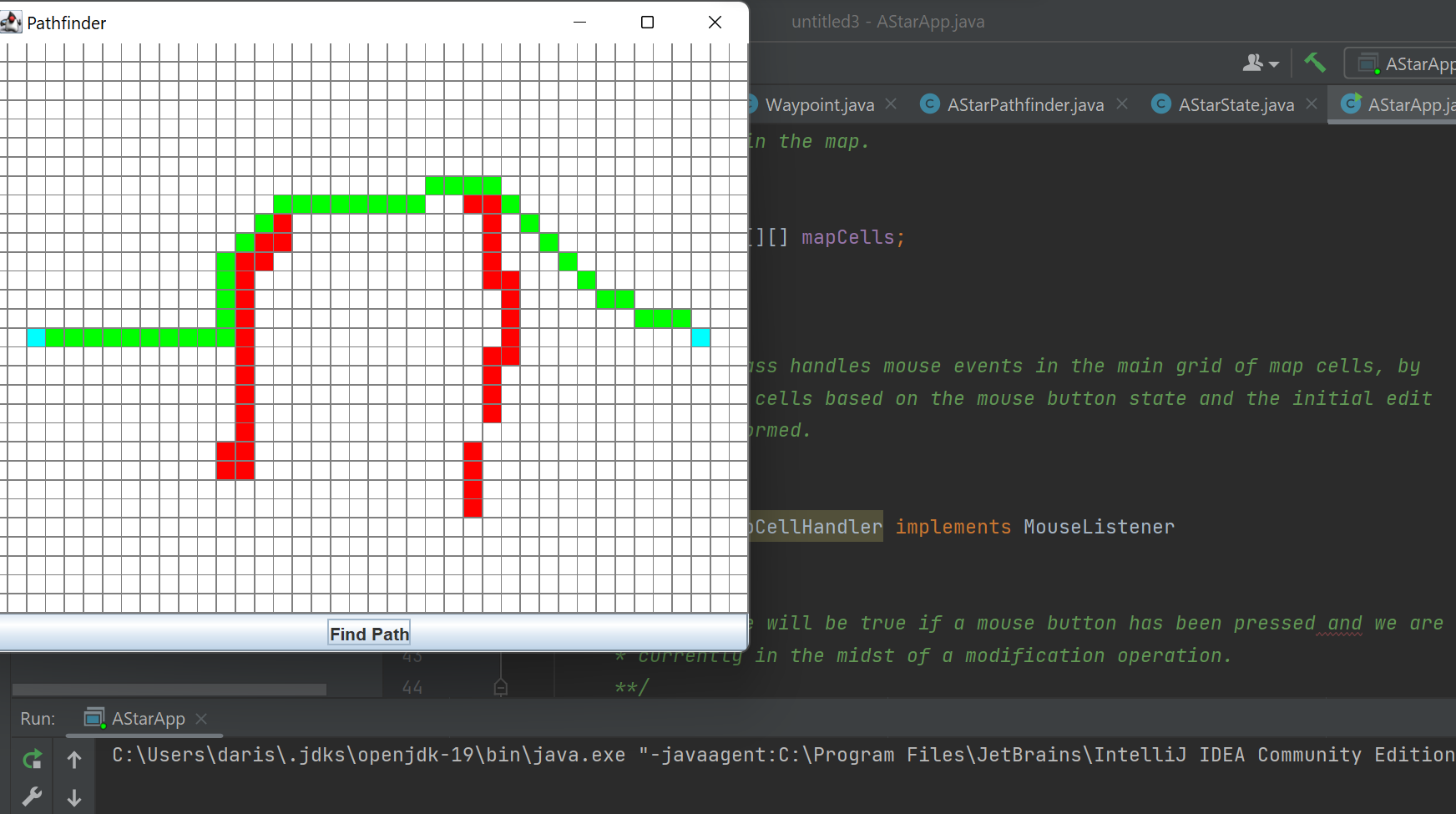
1. Файл AStarState.java

*/\*\*  
 \* This class stores the basic state necessary for the A\* algorithm to compute a  
 \* path across a map. This state includes a collection of "open waypoints" and  
 \* another collection of "closed waypoints." In addition, this class provides  
 \* the basic operations that the A\* pathfinding algorithm needs to perform its  
 \* processing.  
 \*\*/*import java.util.Map;  
import java.util.HashMap;  
public class AStarState  
{  
 */\*\* This is a reference to the map that the A\* algorithm is navigating. \*\*/* private Map2D map;  
 Map<Location, Waypoint> openedWaypoint;  
 Map<Location, Waypoint> closedWaypoint;  
  
 */\*\*  
 \* Initialize a new state object for the A\* pathfinding algorithm to use.  
 \*\*/* public AStarState(Map2D map)  
 {  
 if (map == null)  
 throw new NullPointerException("map cannot be null");  
  
 this.map = map;  
 openedWaypoint = new HashMap<Location, Waypoint>();  
 closedWaypoint = new HashMap<Location, Waypoint>();  
 }  
  
 */\*\* Returns the map that the A\* pathfinder is navigating. \*\*/* public Map2D getMap()  
 {  
 return map;  
 }  
  
 */\*\*  
 \* This method scans through all open waypoints, and returns the waypoint  
 \* with the minimum total cost. If there are no open waypoints, this method  
 \* returns <code>null</code>.  
 \*\*/* public Waypoint getMinOpenWaypoint()  
 {  
 Waypoint minWaypoint = null;  
 //временная переменная  
 Waypoint temp = null;  
 float minPrice = Float.*MAX\_VALUE*;  
 //Проходимся по всем открытым точкам и постепенно находим минимум  
 for (int i=0; i<openedWaypoint.size(); i++) {  
 temp = (Waypoint) openedWaypoint.values().toArray()[i];  
 if (temp.getTotalCost()<minPrice) {  
 minPrice = temp.getTotalCost();  
 minWaypoint = temp;  
 }  
 }  
 return minWaypoint;  
 }  
  
 */\*\*  
 \* This method adds a waypoint to (or potentially updates a waypoint already  
 \* in) the "open waypoints" collection. If there is not already an open  
 \* waypoint at the new waypoint's location then the new waypoint is simply  
 \* added to the collection. However, if there is already a waypoint at the  
 \* new waypoint's location, the new waypoint replaces the old one <em>only  
 \* if</em> the new waypoint's "previous cost" value is less than the current  
 \* waypoint's "previous cost" value.  
 \*\*/* public boolean addOpenWaypoint(Waypoint newWP)  
 {  
 //Пробуем найти точку в открытых точках  
 Waypoint newOpenedWaypoint = openedWaypoint.get(newWP.getLocation());  
 //Если точка не найдена или она есть, но её значение меньше, то кладем ее в открытые  
 if (newOpenedWaypoint==null || newOpenedWaypoint.getPreviousCost()<newWP.getPreviousCost()) {  
 openedWaypoint.put(newWP.getLocation(),newWP);  
 return true;  
 }  
 return false;  
 }  
  
  
 */\*\* Returns the current number of open waypoints. \*\*/* public int numOpenWaypoints()  
 {  
 return openedWaypoint.size();  
 }  
  
  
 */\*\*  
 \* This method moves the waypoint at the specified location from the  
 \* open list to the closed list.  
 \*\*/* public void closeWaypoint(Location loc)  
 {  
 //Получаем точку по хэшу локации из открытых точек  
 Waypoint isOpenedWaypoint = openedWaypoint.get(loc);  
 //Если point = null, то этой точки нет в открытых, т.е. дальше нечего делать  
 if (isOpenedWaypoint == null) return;  
 //Удаляем точку из списка открытых  
 openedWaypoint.remove(loc);  
 //Добавляем точку в список закрытых  
 closedWaypoint.put(loc, isOpenedWaypoint);  
 }  
  
 */\*\*  
 \* Returns true if the collection of closed waypoints contains a waypoint  
 \* for the specified location.  
 \*\*/* public boolean isLocationClosed(Location loc)  
 {  
 return closedWaypoint.get(loc)!=null;  
 }  
}

1. Файл Location.java

*/\*\*  
 \* This class represents a specific location in a 2D map. Coordinates are  
 \* integer values.  
 \*\*/*public class Location  
{  
 */\*\* X coordinate of this location. \*\*/* public int xCoord;  
  
 */\*\* Y coordinate of this location. \*\*/* public int yCoord;  
  
  
 */\*\* Creates a new location with the specified integer coordinates. \*\*/* public Location(int x, int y)  
 {  
 xCoord = x;  
 yCoord = y;  
 }  
  
 */\*\* Creates a new location with coordinates (0, 0). \*\*/* public Location()  
 {  
 this(0, 0);  
 }  
  
 @Override  
 public boolean equals(Object obj) {  
 //получаем ссылку на объект. нам необходимо проверить является ли он экземпляром класса Location  
 if (obj instanceof Location) {  
 Location p = (Location) obj;  
 //Если является - сравниваем его координаты  
 return p.xCoord == xCoord && p.yCoord == yCoord;  
 }  
 //Если не является, возвращаем false  
 return false;  
 }  
  
 @Override  
 public int hashCode() {  
 //Возвращаем хэш-код по его координатам.  
 return xCoord+2+(yCoord-6)\*yCoord;  
 }  
}

**Скриншот выполнения:**



**Ссылка на репозиторий GitHub:** https://github.com/DarinaStepanova/iteh

**Заключение:**

Таким образом, я использовала хранилище java.util.HashMap для каждого из наборов алгоритма А\* с объектами Location в качестве ключей, и объектами Waypoint в качестве значений.