ФЕДЕРАЛЬНОЕ АГЕНСТВО СВЯЗИ

Ордена Трудового Красного Знамени

федеральное государственное бюджетное

образовательное учреждение высшего образования

«Московский Технический Университет Связи и Информатики» (МТУСИ)

Кафедра МКиИТ

Лабораторная работа по технологиям программирования №3

«Java-Я-звезда!»

Выполнил:

Студент 2 курса

Группы БСТ-1602

Пащенко Алексей

Вариант №12

Москва, 2018

Оглавление

[1. Цель и задачи. 3](#_Toc514605219)

[2. Анализ предметной области и выбор инструментария. 3](#_Toc514605220)

[3. Код программы и объяснение функций. 3](#_Toc514605221)

[3.1. AStarState.java. 3](#_Toc514605222)

[3.2. Файл Location 5](#_Toc514605223)

[3.3. Пример работы программы. 6](#_Toc514605224)

[4. Заключение. 7](#_Toc514605225)

1. Цель и задачи.

Изменить классы Location и AStarState, чтобы программа могла найти наикратчайший путь между двумя точками, при этом обходя все заранее указанные препятствия.

1. Анализ предметной области и выбор инструментария.

В текущей работе я использовал бесплатно распространяемый пакет разработчика Java Development Kit. Однако, этот пакет не предусматривает интегрированную среду разработки, поэтому для написания непосредственно кода мною была выбрана программа Visual Studio Code, так как она обеспечивает подсветку интерфейса и автоподстановку текста.

1. Код программы и объяснение функций.
   1. AStarState.java.

import java.util.HashMap;

/\*\*

\* 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.

\*\*/

public class AStarState

{

// Declare the variables needed to navigate the map

private Map2D map;

private HashMap<Location, Waypoint> openPoints;

private HashMap<Location, Waypoint> closePoints;

/\*\*

\* Create a new object of the AStarState class

\*\*/

public AStarState(Map2D map)

{

if (map == null)

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

this.map = map;

openPoints = new HashMap<Location, Waypoint>();

closePoints = new HashMap<Location, Waypoint>();

}

/\*\*

\* Returns the map

\*/

public Map2D getMap()

{

return map;

}

/\*\*

\* This method looks for a point with a minimum value and returns it

\*\*/

public Waypoint getMinOpenWaypoint()

{

        Waypoint[] p=new Waypoint[]{};

        p=openPoints.values().toArray(p);

        Waypoint minPoint=p[0];

        double min=minPoint.getTotalCost();

for(int i=1;i<p.length; i++){

double cost = p[i].getTotalCost();

if (cost<min) {

minPoint=p[i];

min=cost;

}

}

return minPoint;

}

/\*\*

\* This method adds a new point if it has never been added

\*\*/

public boolean addOpenWaypoint(Waypoint newWP)

{

Waypoint p=openPoints.get(newWP.getLocation());

if (p==null){

openPoints.put(newWP.getLocation(),newWP);

return true;

}

if(newWP.getPreviousCost()<p.getPreviousCost()){

openPoints.replace(newWP.getLocation(),newWP);

return true;

}

return false;

}

/\*\*

\* Returns the number of open points

\*/

public int numOpenWaypoints()

{

return openPoints.size();

}

/\*\*

\* Moves a point from open to closed

\*\*/

    public void closeWaypoint(Location loc)

{

Waypoint p = openPoints.remove(loc);

if(p!=null)

{

closePoints.put(loc, p);

}

}

/\*\*

\* Returns true if the collection of closed waypoints contains a waypoint

\* for the specified location.

\*\*/

public boolean isLocationClosed(Location loc)

{

return closePoints.containsKey(loc);

}

}

Класс AStarState является ядром этой программы. Он содержит карту, два списка точек: открытых и закрытых, а также набор функций для оперирования с этими списками. А именно: нахождение минимальной точки, добавление точки в список открытых, перенесение точки из списка открытых в список закрытых, проверка точки на закрытость, возврат карты и возврат количества открытых точек.

* 1. Файл Location

/\*\*

\* 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);

}

public boolean equals(Object obj)

{

Location loc = (Location) obj;

return (xCoord == loc.xCoord && yCoord == loc.yCoord);

}

public int hashCode()

{

int result = 0;

result = xCoord;

result = result\*31+ yCoord;

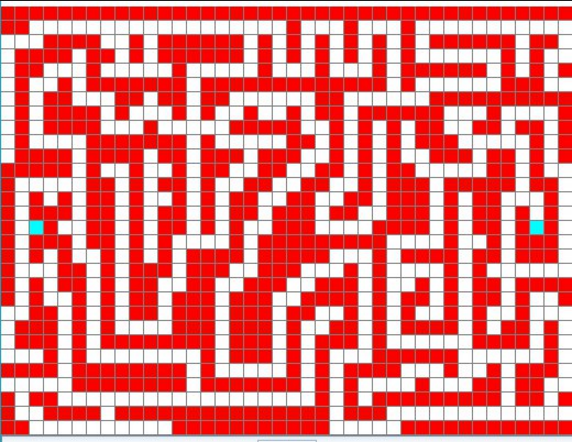
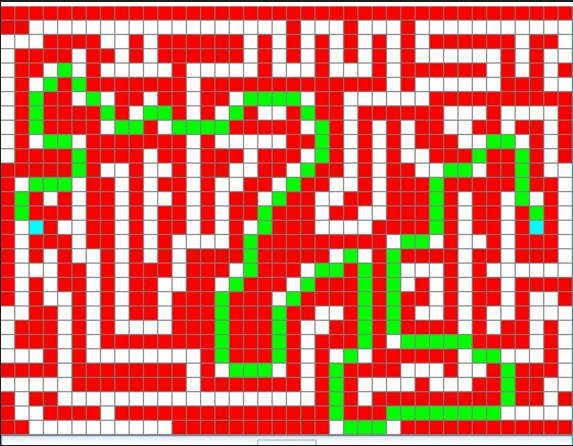
return result;

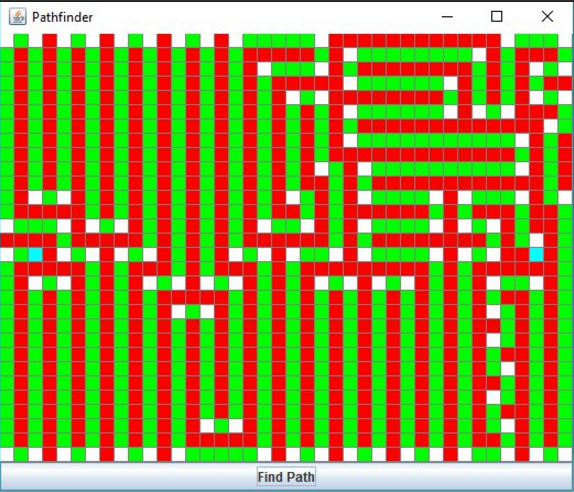
}

}

В этом файле были добавлены две функции: проверка эквивалентности двух объектов equals и получение хеш-кода hashCode.

* 1. Пример работы программы.



1. Заключение.

В ходе работы мною была дописана программа, позволяющая находить кратчайший путь между двумя точками.