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» (МТУСИ)

Кафедра «Математической кибернетики и информационных технологий»

Лабораторная работа №5

По дисциплине: «Java - программирование»

По теме: «Алгоритм A\* («A star»)»

Выполнил студент

Группы БСУ1901:

Иванов Р.А.

Оглавление

[1 Задание 3](#_Toc84964399)

[2 Выполнение лабораторной работы 3](#_Toc84964400)

# 1 Задание

Для начала необходимо подготовить класс Location для совместного использования с классами коллекции Java. Поскольку вы будете использовать контейнеры для хеширования для выполнения данного задания, то для этого необходимо:

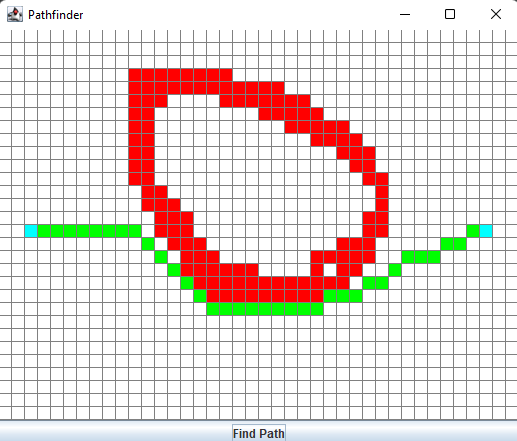
1. Обеспечить реализацию метода equals ().
2. Обеспечить реализацию метода hashcode().

Добавьте два (нестатических) поля в класс AStarState с таким типом, одно для "открытых вершин" и другой для "закрытых вершин". Кроме того, не забудьте инициализировать каждое из этих полей для ссылки на новую пустую коллекцию. После создания и инициализации полей, вы должны реализовать следующие методы в классе AStarState:

1. public int numOpenWaypoints()
2. public Waypoint getMinOpenWaypoint()
3. public boolean addOpenWaypoint(Waypoint newWP)
4. public boolean isLocationClosed(Location loc)
5. public void closeWaypoint(Location loc)

# 2 Выполнение лабораторной работы

На рисунке ниже представлен результат выполнения программы



Ниже представлен код файла Location.java содержащего описание класса point3d:

import java.util.Objects;  
  
*/\*\*  
 \* 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 o) {  
 if (this == o) return true;  
 if (!(o instanceof Location)) return false;  
 Location location = (Location) o;  
 return xCoord == location.xCoord && yCoord == location.yCoord;  
 }  
  
 @Override  
 public int hashCode() {  
 return Objects.*hash*(xCoord, yCoord);  
 }  
}

Ниже представлен код файла .java:

import java.util.HashMap;  
  
  
public class AStarState  
{  
 HashMap<Location, Waypoint> open\_waypoints = new HashMap<Location, Waypoint>();  
 HashMap<Location, Waypoint> closed\_waypoints = new HashMap<Location, Waypoint>();  
 */\*\* This is a reference to the map that the A\* algorithm is navigating. \*\*/* private Map2D map;  
  
  
 */\*\*  
 \* 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;  
 }  
  
 */\*\* 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()  
 {  
 if (numOpenWaypoints() == 0)  
 return null;  
 Waypoint best = null;  
 float best\_cost = Float.*MAX\_VALUE*;  
 for(Location key: open\_waypoints.keySet()){  
 Waypoint waypoint = open\_waypoints.get(key);  
 float waypoint\_total\_cost = waypoint.getTotalCost();  
 if (waypoint\_total\_cost < best\_cost)  
 {  
 best = open\_waypoints.get(key);  
 best\_cost = waypoint\_total\_cost;  
 }  
 }  
  
 return best;  
 }  
  
 */\*\*  
 \* 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)  
 {  
 Location location = newWP.getLocation();  
  
  
 if (open\_waypoints.containsKey(location))  
 {  
  
 Waypoint current\_waypoint = open\_waypoints.get(location);  
 if (newWP.getPreviousCost() < current\_waypoint.getPreviousCost())  
 {  
  
 open\_waypoints.put(location, newWP);  
 return true;  
 }  
  
 return false;  
 }  
  
 open\_waypoints.put(location, newWP);  
 return true;  
 }  
  
  
 */\*\* Returns the current number of open waypoints. \*\*/* public int numOpenWaypoints()  
 {  
 return open\_waypoints.size();  
 }  
  
  
 */\*\*  
 \* This method moves the waypoint at the specified location from the  
 \* open list to the closed list.  
 \*\*/* public void closeWaypoint(Location loc)  
 {  
 Waypoint waypoint = open\_waypoints.remove(loc);  
 closed\_waypoints.put(loc, waypoint);  
 }  
  
 */\*\*  
 \* Returns true if the collection of closed waypoints contains a waypoint  
 \* for the specified location.  
 \*\*/* public boolean isLocationClosed(Location loc)  
 {  
 return closed\_waypoints.containsKey(loc);  
 }  
}