Федеральное агентство связи

Ордена Трудового Красного знамени

Федеральное государственное бюджетное образовательное учреждение

Высшего образования

«Московский технический университет связи и информатики»

Кафедра Математической кибернетики и информационных технологий

Отчет по лабораторной работе №3

по дисциплине «Объектно ориентированное программирование»

на тему «Алгоритм A\*»

Выполнил

Студент группы БФИ1701

Неретин И.Г.

Проверила:

Мосеева М.С.

Москва 2020

1. **Цель работы**

Цель работы – научиться работать с классами HashMap, Set и их методами на языке java.

1. **Задание**

Цель работы определила следующие задачи:

1. Скачать файлы «AStarApp.java», «AStarPathfinder.java», «AStarState.java», «JMapCell.java», «Location.java», «Map2D.java», «Waypoint.java», и включить их в проект;
2. В классе «Location» переопределить методы «equals(object obj)» и «hashCode()»;
3. Создать в классе «AStarState» два поля типа HashMap с ключом типа Location и значением типа Waypoint – набор открытых вершин и закрытых вершин;
4. Разработать в классе «AStarState» следующие методы:
   1. int numOpenWaypoints(), возвращающий число точек в наборе открытых вершин;
   2. Waypoint getMinOpenWaypoint(), возвращающая ссылку на вершину с минимальной общей стоимостью среди всех вершин из набора открытых вершин;
   3. boolean addOpenWaypoint(Waypoint newWP), который добавляет новую вершину в набор открытых вершин, если такой вершины еще нет в наборе или новая вершина имеет меньшую стоимость пути до нее, чем старая (новая вершина лучше, чем старая). Возвращает true, если вершина была успешно добавлена;
   4. boolean isLocationClosed(Location loc), который проверяет, существует ли вершина с таким местоположением в наборе закрытых вершин или нет;
   5. void closeWaypoint(Location loc), который перемещает вершину с таким местоположением из набора открытых вершин в набор закрытых вершин.
5. Скомпилировать и запустить программу, проверить ее работоспособность;
6. Составить отчет о проделанной работе.
7. **Порядок выполнения работы**

Поскольку в процессе выполнения работы был изменен код только классов «AStarState» и «Location», ниже приведен код только этих классов. Остальные классы размещены в Git-репозитории.

Листинг 1 – 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);  }   public boolean equals(Location Loc){  return (Loc.xCoord == this.xCoord)&&(Loc.yCoord == this.yCoord);  }   public int hashCode(){   int result = 17;  int x = xCoord; int y = yCoord;  if (x == 0) x = 37000;  if (y == 0) y = 86500000;  result = 1000 \* result \* xCoord;  result = 1000 \* result \* yCoord;  return result;  } } |

Листинг 2 – AStarState.java

|  |
| --- |
| import java.util.HashMap; import java.util.Iterator;  */\*\*  \* 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 {  */\*\* This is a reference to the map that the A\* algorithm is navigating. \*\*/* private Map2D map;  private HashMap<Location, Waypoint> OpenWaypintMap;  private HashMap<Location, Waypoint> ClosedWaypintMap;   */\*\*  \* 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.OpenWaypintMap = new HashMap<Location, Waypoint>();  this.ClosedWaypintMap = new HashMap<Location, Waypoint>();  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()  {  float minCost = Float.*MAX\_VALUE*;  Waypoint currentPoint;  Waypoint minWaypoint = null;  for (Waypoint waypoint : OpenWaypintMap.values()) {  currentPoint = waypoint;  if (currentPoint.getTotalCost() < minCost) {  minWaypoint = currentPoint;  minCost = currentPoint.getTotalCost();  }  }  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)  {  if (newWP == null) return false;  for (Location currentLocation : OpenWaypintMap.keySet()) {  if (currentLocation.equals(newWP.loc)){  if (OpenWaypintMap.get(currentLocation).getRemainingCost() > newWP.getRemainingCost()){  //новая точка лучше  OpenWaypintMap.put(currentLocation,newWP);  return true;  }  else{  //новая точка хуже  return false;  }  }  }   //в наборе нет такой точки  //System.out.println("add: x = " + newWP.loc.xCoord + " y = " + newWP.loc.yCoord);  OpenWaypintMap.put(newWP.loc,newWP);  return true;  }    */\*\* Returns the current number of open waypoints. \*\*/* public int numOpenWaypoints()  {  return OpenWaypintMap.size();  }    */\*\*  \* This method moves the waypoint at the specified location from the  \* open list to the closed list.  \*\*/* public void closeWaypoint(Location loc)  {  Waypoint contained = OpenWaypintMap.remove(loc);  ClosedWaypintMap.put(loc, contained);  }   */\*\*  \* Returns true if the collection of closed waypoints contains a waypoint  \* for the specified location.  \*\*/* public boolean isLocationClosed(Location loc)  {  return ClosedWaypintMap.containsKey(loc);  } } |

**Результаты работы программы**

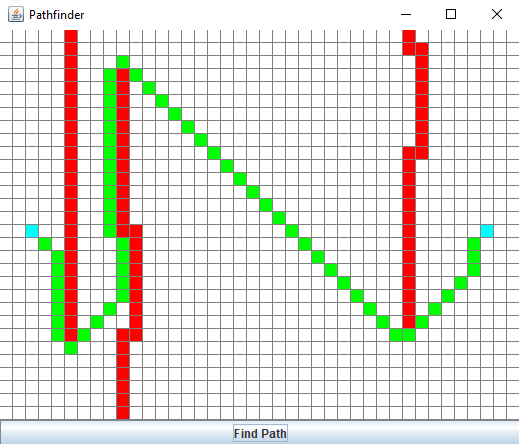


Рисунок 1 – Результат работы программы «Pathfinder» на примере поля №1

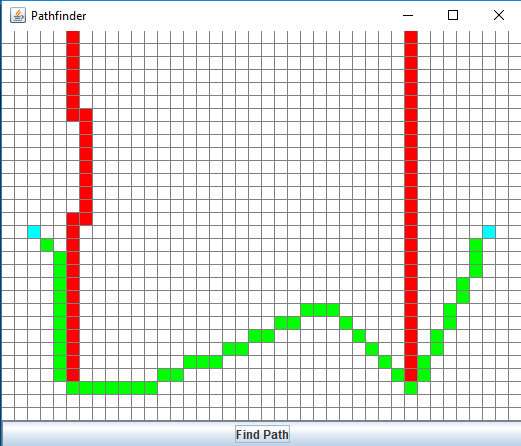


Рисунок 2 – Результат работы программы «Pathfinder» на примере поля №2

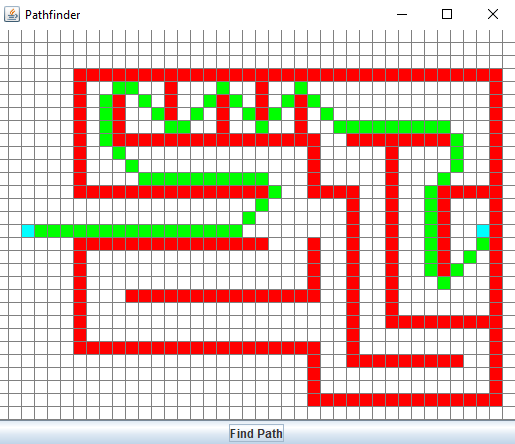


Рисунок 3 – Результат работы программы «Pathfinder» на примере поля №3

1. **Заключение**

В ходе данной работы были изучены классы HashMap и Set языка java. Типы «HashMap» позволяют хранить информацию в виде, требующем быстрого доступа, но не определяющего конкретный порядок, в то время как Set позволяет реализовывать множество, не допускающее повторных значений (коллекция без дубликатов). Эти типы данных позволили реализовать функции работы с алгоритмом А\*. Таким образом, цель данной лабораторной работы была успешно выполнена.

1. **Литература**
2. Камаев В.А., Костерин В.В. Технологии программирования. М.: Высшая школа, 2006;
3. Жоголев Е.А.Технология программирования. – М.: Научный мир, 2004;
4. <https://www.google.ru/>