**Федеральное агентство связи**

**Ордена Трудового Красного Знамени**

**Федеральное государственное бюджетное образовательное учреждение высшего образования**

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

**Кафедра Информатики**



**Отчет по лабораторной работе №3**

по предмету «КТП»

Выполнил: студент группы БВТ1802

Ткачев Александр Владимирович

Руководитель:

Ксения Андреевна Полянцева

Москва 2020

**1 Цель работы**

Цель работы: изучить алгоритм A\*.

**2 Задание**

Дополнить исходный текст программы таким образом, чтобы она находила кратчайший путь в обход препятствий.

**3 Текст программы**

**Class AStarApp**

package com.company;  
import java.awt.\*;  
import java.awt.event.\*;  
import javax.swing.\*;  
  
public class AStarApp {  
  
 */\*\* The number of grid cells in the X direction. \*\*/* private int width;  
  
 */\*\* The number of grid cells in the Y direction. \*\*/* private int height;  
  
 */\*\* The location where the path starts from. \*\*/* private Location startLoc;  
  
 */\*\* The location where the path is supposed to finish. \*\*/* private Location finishLoc;  
  
 */\*\*  
 \* This is a 2D array of UI components that provide display and manipulation  
 \* of the cells in the map.  
 \*\*\*/* private JMapCell[][] mapCells;  
  
  
 */\*\*  
 \* This inner class handles mouse events in the main grid of map cells, by  
 \* modifying the cells based on the mouse button state and the initial edit  
 \* that was performed.  
 \*\*/* private class MapCellHandler implements MouseListener  
 {  
 */\*\*  
 \* This value will be true if a mouse button has been pressed and we are  
 \* currently in the midst of a modification operation.  
 \*\*/* private boolean modifying;  
  
 */\*\*  
 \* This value records whether we are making cells passable or  
 \* impassable. Which it is depends on the original state of the cell  
 \* that the operation was started within.  
 \*\*/* private boolean makePassable;  
  
 */\*\* Initiates the modification operation. \*\*/* public void mousePressed(MouseEvent e)  
 {  
 modifying = true;  
  
 JMapCell cell = (JMapCell) e.getSource();  
  
 // If the current cell is passable then we are making them  
 // impassable; if it's impassable then we are making them passable.  
  
 makePassable = !cell.isPassable();  
  
 cell.setPassable(makePassable);  
 }  
  
 */\*\* Ends the modification operation. \*\*/* public void mouseReleased(MouseEvent e)  
 {  
 modifying = false;  
 }  
  
 */\*\*  
 \* If the mouse has been pressed, this continues the modification  
 \* operation into the new cell.  
 \*\*/* public void mouseEntered(MouseEvent e)  
 {  
 if (modifying)  
 {  
 JMapCell cell = (JMapCell) e.getSource();  
 cell.setPassable(makePassable);  
 }  
 }  
  
 */\*\* Not needed for this handler. \*\*/* public void mouseExited(MouseEvent e)  
 {  
 // This one we ignore.  
 }  
  
 */\*\* Not needed for this handler. \*\*/* public void mouseClicked(MouseEvent e)  
 {  
 // And this one too.  
 }  
 }  
  
  
 */\*\*  
 \* Creates a new instance of AStarApp with the specified map width and  
 \* height.  
 \*\*/* public AStarApp(int w, int h) {  
 if (w <= 0)  
 throw new IllegalArgumentException("w must be > 0; got " + w);  
  
 if (h <= 0)  
 throw new IllegalArgumentException("h must be > 0; got " + h);  
  
 width = w;  
 height = h;  
  
 startLoc = new Location(2, h / 2);  
 finishLoc = new Location(w - 3, h / 2);  
 }  
  
  
 */\*\*  
 \* Simple helper method to set up the Swing user interface. This is called  
 \* from the Swing event-handler thread to be threadsafe.  
 \*\*/* private void initGUI()  
 {  
 JFrame frame = new JFrame("Pathfinder");  
 frame.setDefaultCloseOperation(JFrame.*EXIT\_ON\_CLOSE*);  
 Container contentPane = frame.getContentPane();  
  
 contentPane.setLayout(new BorderLayout());  
  
 // Use GridBagLayout because it actually respects the preferred size  
 // specified by the components it lays out.  
  
 GridBagLayout gbLayout = new GridBagLayout();  
 GridBagConstraints gbConstraints = new GridBagConstraints();  
 gbConstraints.fill = GridBagConstraints.*BOTH*;  
 gbConstraints.weightx = 1;  
 gbConstraints.weighty = 1;  
 gbConstraints.insets.set(0, 0, 1, 1);  
  
 JPanel mapPanel = new JPanel(gbLayout);  
 mapPanel.setBackground(Color.*GRAY*);  
  
 mapCells = new JMapCell[width][height];  
  
 MapCellHandler cellHandler = new MapCellHandler();  
  
 for (int y = 0; y < height; y++)  
 {  
 for (int x = 0; x < width; x++)  
 {  
 mapCells[x][y] = new JMapCell();  
  
 gbConstraints.gridx = x;  
 gbConstraints.gridy = y;  
  
 gbLayout.setConstraints(mapCells[x][y], gbConstraints);  
  
 mapPanel.add(mapCells[x][y]);  
 mapCells[x][y].addMouseListener(cellHandler);  
 }  
 }  
  
 contentPane.add(mapPanel, BorderLayout.*CENTER*);  
  
 JButton findPathButton = new JButton("Find Path");  
 findPathButton.addActionListener(new ActionListener() {  
 public void actionPerformed(ActionEvent e) { findAndShowPath(); }  
 });  
  
 contentPane.add(findPathButton, BorderLayout.*SOUTH*);  
  
 frame.pack();  
 frame.setVisible(true);  
  
 mapCells[startLoc.xCoord][startLoc.yCoord].setEndpoint(true);  
 mapCells[finishLoc.xCoord][finishLoc.yCoord].setEndpoint(true);  
 }  
  
  
 */\*\* Kicks off the application. Called from the {****@link*** *#main} method. \*\*/* private void start()  
 {  
 SwingUtilities.*invokeLater*(new Runnable() {  
 public void run() { initGUI(); }  
 });  
 }  
  
  
 */\*\*  
 \* This helper method attempts to compute a path using the current map  
 \* state. The implementation is rather slow; a new {****@link*** *Map2D} object is  
 \* created, and initialized from the current application state. Then the A\*  
 \* pathfinder is called, and if a path is found, the display is updated to  
 \* show the path that was found. (A better solution would use the Model  
 \* View Controller design pattern.)  
 \*\*/* private void findAndShowPath()  
 {  
 // Create a Map2D object containing the current state of the user input.  
  
 Map2D map = new Map2D(width, height);  
 map.setStart(startLoc);  
 map.setFinish(finishLoc);  
  
 for (int y = 0; y < height; y++)  
 {  
 for (int x = 0; x < width; x++)  
 {  
 mapCells[x][y].setPath(false);  
  
 if (mapCells[x][y].isPassable())  
 map.setCellValue(x, y, 0);  
 else  
 map.setCellValue(x, y, Integer.*MAX\_VALUE*);  
 }  
 }  
  
 // Try to compute a path. If one can be computed, mark all cells in the  
 // path.  
  
 Waypoint wp = AStarPathfinder.*computePath*(map);  
  
 while (wp != null)  
 {  
 Location loc = wp.getLocation();  
 mapCells[loc.xCoord][loc.yCoord].setPath(true);  
  
 wp = wp.getPrevious();  
 }  
 }  
  
  
 */\*\*  
 \* Entry-point for the application. No command-line arguments are  
 \* recognized at this time.  
 \*\*/* public static void main(String[] args) {  
 AStarApp app = new AStarApp(40, 30);  
 app.start();  
 }  
}

**Class AStarState**

package com.company;  
  
import java.util.HashMap;  
  
*/\*\*  
 \* Класс хранит базовое состояние необходимое для алгоритма А\*, чтобы вычислить  
 \* путь по карте. Это состояние включает коллекции "открытых точек" и другие коллеции  
 \* "закрытых точек". Также этот класс обеспечивает основные операци А\* алгоритма,  
 \* для обработки алгоритма.  
 \*/*public class AStarState  
{  
 */\*\* Это ссылка на карту, по которой работает алгоритм А\*. \*\*/* private Map2D map;  
 public HashMap <Location, Waypoint> Opened = new HashMap <Location, Waypoint>();  
 public HashMap <Location, Waypoint> Closed = new HashMap <Location, Waypoint>();  
  
 */\*\*  
 \* Создание нового объекта для алгоритма поиска А\*.  
 \*\*/* public AStarState(Map2D map)  
 {  
 if (map == null)  
 throw new NullPointerException("map cannot be null");  
  
 this.map = map;  
 }  
  
 */\*\* Возвращает карту, по которой работает алгоритм А\*. \*\*/* public Map2D getMap()  
 {  
 return map;  
 }  
  
 */\*\*  
 \* Этот метод просматривает все открытые точки и возвращает точку с  
 \* минимальным общим значением.  
 \* Если нет открытых точек, метод возвращает null.  
 \*/* public Waypoint getMinOpenWaypoint()  
 {  
 if (Opened.size() == 0) return null;  
  
 Waypoint first = (Waypoint) Opened.values().toArray()[0];  
 float min = first.getTotalCost();  
  
 for(int i = 0; i < Opened.size();i++)  
 {  
 Waypoint wp1 = (Waypoint) Opened.values().toArray()[i];  
  
 if(min >= wp1.getTotalCost())  
 {  
 min = wp1.getTotalCost();  
 first = wp1;  
 }  
 }  
 return first;  
 }  
  
 */\*\*  
 \* Этот метод добавляет точку в коллекцию открытых точек.  
 \* Если в местополодении новой точки нет уде открытой новой точки,  
 \* то она добавляется в коллекцию. Однако, если в коллекции есть точка,  
 \* новая точка заменяет старую точку, если значение новой точки меньше  
 \* текущей.  
 \*/* public boolean addOpenWaypoint(Waypoint newWP)  
 {  
 if(!Opened.containsKey(newWP.getLocation())) {  
 Opened.put(newWP.getLocation(), newWP);  
 return true;  
 }  
 else  
 {  
 if(Opened.get(newWP.getLocation()).getRemainingCost() > newWP.getRemainingCost())  
 {  
 Opened.put(newWP.getLocation(), newWP);  
 return true;  
 }  
 }  
 return false;  
 }  
  
  
 */\*\* Возвраащет количество открытых точек. \*\*/* public int numOpenWaypoints()  
 {  
 return Opened.size();  
 }  
  
  
 */\*\*  
 \* Этот метод перемещает точку в указанном месте из открытого списка в закрытый.  
 \* из открытого списка в закрытый.  
 \*\*/* public void closeWaypoint(Location loc)  
 {  
 Waypoint wp = Opened.get(loc);  
 Opened.remove(loc);  
 Closed.put(loc, wp);  
 }  
  
 */\*\*  
 \* Возвращает true, если коллекция зыкрытых точек содержит точку  
 \* для указанного местоположения.  
 \*\*/* public boolean isLocationClosed(Location loc)  
 {  
 if (Closed.containsKey(loc)) return true;  
 return false;  
 }  
}

**Class Location**

package com.company;  
  
import java.util.Objects;  
  
public class Location  
{  
 */\*\* Координаты по X. \*\*/* public int xCoord;  
  
 */\*\* Координаты по Y. \*\*/* public int yCoord;  
  
  
 */\*\* Создание нового местоположения с указанными целочисленными координатами. \*\*/* public Location(int x, int y)  
 {  
 xCoord = x;  
 yCoord = y;  
 }  
  
 */\*\* Создание местоположения с координатами (0,0). \*\*/* public Location()  
 {  
 this(0, 0);  
 }  
  
 @Override  
 public int hashCode() {  
 return Objects.*hash*(xCoord,yCoord);  
 }  
  
 @Override  
 public boolean equals(Object coord) {  
 Location lock = (Location) coord;  
 if (this.xCoord == lock.xCoord && this.yCoord == lock.yCoord) return true;  
 return false;  
 }  
}

**4 Работа программы**

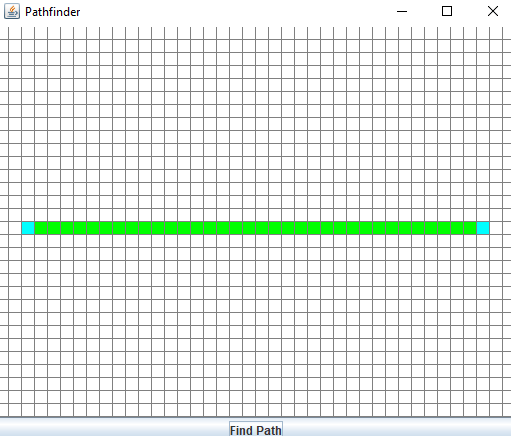


Рисунок 1 — Результат работы программы