package game;

import java.awt.Color;

import java.awt.Graphics2D;

import java.awt.Image;

import java.awt.Toolkit;

import java.awt.geom.AffineTransform;

import java.io.BufferedReader;

import java.io.IOException;

import java.io.InputStream;

import java.io.InputStreamReader;

public class Map {

private int[][][][] map;

private int mapSecX;

private int mapSecY;

private int tileSize;

private Image water;

private Image waterBorderR;

private Image waterBorderL;

private Image waterBorderU;

private Image waterBorderUL;

private Image waterBorderUR;

private Image waterBorderD;

private Image waterBorderDL;

private Image waterBorderDR;

private Image islandC;

private Image islandL;

private Image islandR;

private Image islandU;

private Image islandD;

private Image islandUL;

private Image islandDL;

private Image islandUR;

private Image islandDR;

private Image islandIUL;

private Image islandIUR;

private Image islandIDL;

private Image islandIDR;

private Image GislandC;

private Image GislandL;

private Image GislandR;

private Image GislandU;

private Image GislandD;

private Image GislandUL;

private Image GislandDL;

private Image GislandUR;

private Image GislandDR;

private Image GislandIUL;

private Image GislandIUR;

private Image GislandIDL;

private Image GislandIDR;

private Image sidebarPic;

private Image pointer;

private Image shopText;

public Map() {

mapSecX = 0;

mapSecY = 0;

tileSize = DS.tileSize;

map = loadGameMap("/GameFiles/MapData.txt");

water = loadImage("/GameFiles/Tiles/tile\_73.png");

waterBorderR = loadImage("/GameFiles/Tiles/tile\_26.png");

waterBorderL = loadImage("/GameFiles/Tiles/tile\_28.png");

waterBorderD = loadImage("/GameFiles/Tiles/tile\_11.png");

waterBorderDR = loadImage("/GameFiles/Tiles/tile\_75.png");

waterBorderDL = loadImage("/GameFiles/Tiles/tile\_74.png");

waterBorderU = loadImage("/GameFiles/Tiles/tile\_43.png");

waterBorderUR = loadImage("/GameFiles/Tiles/tile\_59.png");

waterBorderUL = loadImage("/GameFiles/Tiles/tile\_58.png");

islandC = loadImage("/GameFiles/Tiles/tile\_69.png");

islandL = loadImage("/GameFiles/Tiles/tile\_17.png");

islandR = loadImage("/GameFiles/Tiles/tile\_19.png");

islandU = loadImage("/GameFiles/Tiles/tile\_02.png");

islandD = loadImage("/GameFiles/Tiles/tile\_34.png");

islandUL = loadImage("/GameFiles/Tiles/tile\_01.png");

islandDL = loadImage("/GameFiles/Tiles/tile\_33.png");

islandUR = loadImage("/GameFiles/Tiles/tile\_03.png");

islandDR = loadImage("/GameFiles/Tiles/tile\_35.png");

islandIUL = loadImage("/GameFiles/Tiles/tile\_04.png");

islandIDL = loadImage("/GameFiles/Tiles/tile\_20.png");

islandIUR = loadImage("/GameFiles/Tiles/tile\_05.png");

islandIDR = loadImage("/GameFiles/Tiles/tile\_21.png");

GislandC = loadImage("/GameFiles/Tiles/tile\_23.png");

GislandL = loadImage("/GameFiles/Tiles/tile\_22.png");

GislandR = loadImage("/GameFiles/Tiles/tile\_25.png");

GislandU = loadImage("/GameFiles/Tiles/tile\_07.png");

GislandD = loadImage("/GameFiles/Tiles/tile\_55.png");

GislandUL = loadImage("/GameFiles/Tiles/tile\_06.png");

GislandDL = loadImage("/GameFiles/Tiles/tile\_54.png");

GislandUR = loadImage("/GameFiles/Tiles/tile\_09.png");

GislandDR = loadImage("/GameFiles/Tiles/tile\_57.png");

GislandIUL = loadImage("/GameFiles/Tiles/tile\_36.png");

GislandIDL = loadImage("/GameFiles/Tiles/tile\_52.png");

GislandIUR = loadImage("/GameFiles/Tiles/tile\_37.png");

GislandIDR = loadImage("/GameFiles/Tiles/tile\_53.png");

sidebarPic = loadImage("/GameFiles/Sidebar.png");

pointer = loadImage("/GameFiles/Pointer.png");

shopText = loadImage("/GameFiles/ShopText.png");

}

public void drawMap(Graphics2D g) {

g.setPaint(Color.black);

g.fillRect(0, 0, DS.screenWidth, DS.screenHeight);

//main map

for (int i = 0; i < 14; i++) {

for (int j = 0; j < 10; j++) {

g.drawImage(water, i \* tileSize, j \* tileSize + DS.gap, tileSize, tileSize, null);

if (map[mapSecX][mapSecY][i][j] == 1) {

g.drawImage(waterBorderL, i \* tileSize, j \* tileSize + DS.gap, tileSize, tileSize, null);

} else if (map[mapSecX][mapSecY][i][j] == 2) {

g.drawImage(waterBorderR, i \* tileSize, j \* tileSize + DS.gap, tileSize, tileSize, null);

} else if (map[mapSecX][mapSecY][i][j] == 3) {

g.drawImage(waterBorderU, i \* tileSize, j \* tileSize + DS.gap, tileSize, tileSize, null);

} else if (map[mapSecX][mapSecY][i][j] == 4) {

g.drawImage(waterBorderD, i \* tileSize, j \* tileSize + DS.gap, tileSize, tileSize, null);

} else if (map[mapSecX][mapSecY][i][j] == 5) {

g.drawImage(waterBorderUL, i \* tileSize, j \* tileSize + DS.gap, tileSize, tileSize, null);

} else if (map[mapSecX][mapSecY][i][j] == 6) {

g.drawImage(waterBorderDL, i \* tileSize, j \* tileSize + DS.gap, tileSize, tileSize, null);

} else if (map[mapSecX][mapSecY][i][j] == 7) {

g.drawImage(waterBorderUR, i \* tileSize, j \* tileSize + DS.gap, tileSize, tileSize, null);

} else if (map[mapSecX][mapSecY][i][j] == 8) {

g.drawImage(waterBorderDR, i \* tileSize, j \* tileSize + DS.gap, tileSize, tileSize, null);

} else if (map[mapSecX][mapSecY][i][j] == 9) {

g.drawImage(islandC, i \* tileSize, j \* tileSize + DS.gap, tileSize, tileSize, null);

} else if (map[mapSecX][mapSecY][i][j] == 10) {

g.drawImage(islandL, i \* tileSize, j \* tileSize + DS.gap, tileSize, tileSize, null);

} else if (map[mapSecX][mapSecY][i][j] == 11) {

g.drawImage(islandR, i \* tileSize, j \* tileSize + DS.gap, tileSize, tileSize, null);

} else if (map[mapSecX][mapSecY][i][j] == 12) {

g.drawImage(islandU, i \* tileSize, j \* tileSize + DS.gap, tileSize, tileSize, null);

} else if (map[mapSecX][mapSecY][i][j] == 13) {

g.drawImage(islandD, i \* tileSize, j \* tileSize + DS.gap, tileSize, tileSize, null);

} else if (map[mapSecX][mapSecY][i][j] == 14) {

g.drawImage(islandUL, i \* tileSize, j \* tileSize + DS.gap, tileSize, tileSize, null);

} else if (map[mapSecX][mapSecY][i][j] == 15) {

g.drawImage(islandDL, i \* tileSize, j \* tileSize + DS.gap, tileSize, tileSize, null);

} else if (map[mapSecX][mapSecY][i][j] == 16) {

g.drawImage(islandUR, i \* tileSize, j \* tileSize + DS.gap, tileSize, tileSize, null);

} else if (map[mapSecX][mapSecY][i][j] == 17) {

g.drawImage(islandDR, i \* tileSize, j \* tileSize + DS.gap, tileSize, tileSize, null);

} else if (map[mapSecX][mapSecY][i][j] == 18) {

g.drawImage(islandIUL, i \* tileSize, j \* tileSize + DS.gap, tileSize, tileSize, null);

} else if (map[mapSecX][mapSecY][i][j] == 19) {

g.drawImage(islandIDL, i \* tileSize, j \* tileSize + DS.gap, tileSize, tileSize, null);

} else if (map[mapSecX][mapSecY][i][j] == 20) {

g.drawImage(islandIUR, i \* tileSize, j \* tileSize + DS.gap, tileSize, tileSize, null);

} else if (map[mapSecX][mapSecY][i][j] == 21) {

g.drawImage(islandIDR, i \* tileSize, j \* tileSize + DS.gap, tileSize, tileSize, null);

} else if (map[mapSecX][mapSecY][i][j] == 22) {

g.drawImage(GislandC, i \* tileSize, j \* tileSize + DS.gap, tileSize, tileSize, null);

} else if (map[mapSecX][mapSecY][i][j] == 23) {

g.drawImage(GislandL, i \* tileSize, j \* tileSize + DS.gap, tileSize, tileSize, null);

} else if (map[mapSecX][mapSecY][i][j] == 24) {

g.drawImage(GislandR, i \* tileSize, j \* tileSize + DS.gap, tileSize, tileSize, null);

} else if (map[mapSecX][mapSecY][i][j] == 25) {

g.drawImage(GislandU, i \* tileSize, j \* tileSize + DS.gap, tileSize, tileSize, null);

} else if (map[mapSecX][mapSecY][i][j] == 26) {

g.drawImage(GislandD, i \* tileSize, j \* tileSize + DS.gap, tileSize, tileSize, null);

} else if (map[mapSecX][mapSecY][i][j] == 27) {

g.drawImage(GislandUL, i \* tileSize, j \* tileSize + DS.gap, tileSize, tileSize, null);

} else if (map[mapSecX][mapSecY][i][j] == 28) {

g.drawImage(GislandDL, i \* tileSize, j \* tileSize + DS.gap, tileSize, tileSize, null);

} else if (map[mapSecX][mapSecY][i][j] == 29) {

g.drawImage(GislandUR, i \* tileSize, j \* tileSize + DS.gap, tileSize, tileSize, null);

} else if (map[mapSecX][mapSecY][i][j] == 30) {

g.drawImage(GislandDR, i \* tileSize, j \* tileSize + DS.gap, tileSize, tileSize, null);

} else if (map[mapSecX][mapSecY][i][j] == 31) {

g.drawImage(GislandIUL, i \* tileSize, j \* tileSize + DS.gap, tileSize, tileSize, null);

} else if (map[mapSecX][mapSecY][i][j] == 32) {

g.drawImage(GislandIDL, i \* tileSize, j \* tileSize + DS.gap, tileSize, tileSize, null);

} else if (map[mapSecX][mapSecY][i][j] == 33) {

g.drawImage(GislandIUR, i \* tileSize, j \* tileSize + DS.gap, tileSize, tileSize, null);

} else if (map[mapSecX][mapSecY][i][j] == 34) {

g.drawImage(GislandIDR, i \* tileSize, j \* tileSize + DS.gap, tileSize, tileSize, null);

}

}

}

}

public void drawShopText(Graphics2D g){

AffineTransform shopTextTrans = new AffineTransform();

shopTextTrans.translate(DS.tileSize\*4, DS.gap+DS.tileSize\*5.75+DS.scale\*20);

shopTextTrans.scale(DS.scale, DS.scale);

//shopTextTrans.translate(-shopText.getWidth(null) / 2, -shopText.getHeight(null) / 2);

g.drawImage(shopText, shopTextTrans, null);

}

public void drawSidebar(Graphics2D g) {

AffineTransform sidebarTrans = new AffineTransform();

sidebarTrans.translate(DS.gameWidth, DS.gap);

sidebarTrans.scale(DS.scale, DS.scale);

g.drawImage(sidebarPic, sidebarTrans, null);

//minimap

for (int bigX = 0; bigX < 5; bigX++) {

for (int bigY = 0; bigY < 5; bigY++) {

for (int i = 0; i < 14; i++) {

for (int j = 0; j < 10; j++) {

AffineTransform tile = new AffineTransform();

tile.translate(DS.scale \* (i \* 8 + bigX \* 112 + 48) + DS.gameWidth, DS.scale \* (j \* 8 + bigY \* 80 + 41) + DS.gap);

tile.scale(DS.scale \* 8 / 128, DS.scale \* 8 / 128);

g.drawImage(water, tile, null);

if (map[bigX][bigY][i][j] == 1) {

g.drawImage(waterBorderL, tile, null);

} else if (map[bigX][bigY][i][j] == 2) {

g.drawImage(waterBorderR, tile, null);

} else if (map[bigX][bigY][i][j] == 3) {

g.drawImage(waterBorderU, tile, null);

} else if (map[bigX][bigY][i][j] == 4) {

g.drawImage(waterBorderD, tile, null);

} else if (map[bigX][bigY][i][j] == 5) {

g.drawImage(waterBorderUL, tile, null);

} else if (map[bigX][bigY][i][j] == 6) {

g.drawImage(waterBorderDL, tile, null);

} else if (map[bigX][bigY][i][j] == 7) {

g.drawImage(waterBorderUR, tile, null);

} else if (map[bigX][bigY][i][j] == 8) {

g.drawImage(waterBorderDR, tile, null);

} else if (map[bigX][bigY][i][j] == 9) {

g.drawImage(islandC, tile, null);

} else if (map[bigX][bigY][i][j] == 10) {

g.drawImage(islandL, tile, null);

} else if (map[bigX][bigY][i][j] == 11) {

g.drawImage(islandR, tile, null);

} else if (map[bigX][bigY][i][j] == 12) {

g.drawImage(islandU, tile, null);

} else if (map[bigX][bigY][i][j] == 13) {

g.drawImage(islandD, tile, null);

} else if (map[bigX][bigY][i][j] == 14) {

g.drawImage(islandUL, tile, null);

} else if (map[bigX][bigY][i][j] == 15) {

g.drawImage(islandDL, tile, null);

} else if (map[bigX][bigY][i][j] == 16) {

g.drawImage(islandUR, tile, null);

} else if (map[bigX][bigY][i][j] == 17) {

g.drawImage(islandDR, tile, null);

} else if (map[bigX][bigY][i][j] == 18) {

g.drawImage(islandIUL, tile, null);

} else if (map[bigX][bigY][i][j] == 19) {

g.drawImage(islandIDL, tile, null);

} else if (map[bigX][bigY][i][j] == 20) {

g.drawImage(islandIUR, tile, null);

} else if (map[bigX][bigY][i][j] == 21) {

g.drawImage(islandIDR, tile, null);

} else if (map[bigX][bigY][i][j] == 22) {

g.drawImage(GislandC, tile, null);

} else if (map[bigX][bigY][i][j] == 23) {

g.drawImage(GislandL, tile, null);

} else if (map[bigX][bigY][i][j] == 24) {

g.drawImage(GislandR, tile, null);

} else if (map[bigX][bigY][i][j] == 25) {

g.drawImage(GislandU, tile, null);

} else if (map[bigX][bigY][i][j] == 26) {

g.drawImage(GislandD, tile, null);

} else if (map[bigX][bigY][i][j] == 27) {

g.drawImage(GislandUL, tile, null);

} else if (map[bigX][bigY][i][j] == 28) {

g.drawImage(GislandDL, tile, null);

} else if (map[bigX][bigY][i][j] == 29) {

g.drawImage(islandUR, tile, null);

} else if (map[bigX][bigY][i][j] == 30) {

g.drawImage(GislandDR, tile, null);

} else if (map[bigX][bigY][i][j] == 31) {

g.drawImage(GislandIUL, tile, null);

} else if (map[bigX][bigY][i][j] == 32) {

g.drawImage(GislandIDL, tile, null);

} else if (map[bigX][bigY][i][j] == 33) {

g.drawImage(GislandIUR, tile, null);

} else if (map[bigX][bigY][i][j] == 34) {

g.drawImage(GislandIDR, tile, null);

}

}

}

}

}

AffineTransform pointerTrans = new AffineTransform();

pointerTrans.translate(DS.gameWidth + DS.scale \* (48 + mapSecX \* 112 + 56), DS.gap + DS.scale \* (41 + mapSecY \* 80 + 40));

pointerTrans.scale(DS.scale, DS.scale);

pointerTrans.translate(-pointer.getWidth(null) / 2, -pointer.getHeight(null) / 2);

g.drawImage(pointer, pointerTrans, null);

}

public Image loadImage(String src) {

Image im = null;

try {

im = Toolkit.getDefaultToolkit().getImage(getClass().getResource(src));

} catch (Exception e) {

}

return im;

}

public int getTile(int tX, int tY) {

return map[mapSecX][mapSecY][tX][tY];

}

private int[][][][] loadGameMap(String src) {

int count = 0;

int[][][][] temp;

try {

InputStream in = getClass().getResourceAsStream(src);

BufferedReader br = new BufferedReader(new InputStreamReader(in));

while (br.readLine() != null) {

for (int i = 0; i < 11; i++) {

br.readLine();

}

count++;

}

br.close();

} catch (IOException e) {

System.out.println("Error: " + e);

}

temp = new int[5][5][14][10];

int num1, num2;

String[] line;

try {

InputStream in = getClass().getResourceAsStream(src);

BufferedReader br = new BufferedReader(new InputStreamReader(in));

for (int i = 0; i < count; i++) {

num1 = Integer.parseInt(br.readLine());

num2 = Integer.parseInt(br.readLine());

for (int j = 0; j < 10; j++) {

line = br.readLine().split(" ");

for (int k = 0; k < 14; k++) {

temp[num1][num2][k][j] = Integer.parseInt(line[k]);

}

}

}

} catch (IOException e) {

System.out.println("Error: " + e);

}

return temp;

}

public int getMapSecX() {

return mapSecX;

}

public int getMapSecY() {

return mapSecY;

}

public int getMapSecXSize() {

return map.length;

}

public int getMapSecYSize() {

return map[0].length;

}

public void setMapSecX(int val) {

mapSecX = val;

}

public void setMapSecY(int val) {

mapSecY = val;

}

}