}

Assignment 3
Code Appendix

```
public class Building {

        int height;
        int width;
        int centerOfMassX;
        int centerOfMassY;
        int area;
        int buildingNumber;
        String buildingName;

        public Building(int buildNum, int buildArea){
                //TODO
                buildingNumber = buildNum;
                area = buildArea;
        }

        public void setHeight(int newHeight){
                height = newHeight;
        }

        public int getHeight(){
                return height;
        }

        public void setWidth(int newWidth){
                width = newWidth;
        }

        public int getWidth(){
                return width;
        }

        public void setCenterOfMassX(int newCenterOfMassX){
                centerOfMassX = newCenterOfMassX;
        }

        public int getCenterOfMassX(){
                return centerOfMassX;
        }

        public void setCenterOfMassY(int newCenterOfMassY){
                centerOfMassY = newCenterOfMassY;
        }

        public int getCenterOfMassY(){
                return centerOfMassY;
        }
```

```
	public void setArea(int newArea){
		area = newArea;
	}

	public int getArea(){
		return area;
	}

	public void setBuildingNumber(int newBuildingNumber){
		buildingNumber = newBuildingNumber;
	}

	public int getBuildingNumber(){
		return buildingNumber;
	}

	public void setBuildingName(String newBuildingName){
		buildingName = newBuildingName;
	}

	public String getBuildingName(){
		return buildingName;
	}
}
```

```
import org.opencv.core.Core;

import java.io.*;
import java.text.DecimalFormat;
import java.util.*;

import org.opencv.core.CvType;
import org.opencv.core.Mat;
import org.opencv.core.MatOfFloat;
import org.opencv.core.MatOfInt;
import org.opencv.core.MatOfPoint;
import org.opencv.core.MatOfPoint2f;
import org.opencv.core.Point;
import org.opencv.core.Rect;
import org.opencv.core.Scalar;
import org.opencv.core.Size;
import org.opencv.highgui.Highgui;
import org.opencv.imgproc.Imgproc;
import org.opencv.imgproc.Moments;

public class WhatDescriptions {
```

```
private int magic_small = 582; //TODO Magic number for small area
private int magic_large = 4400; //TODO Magic number for large area
private int magic_largest; // Corresponds to the integer value of the building
private int magic_smallest; //Corresponds to the integer value of the building;
private int smallest;
private int largest;
private List<MatOfPoint> globContours;
Building[] globBuildings;

public void run(){
        //TODO
        int WIDTH = 275;
        int HEIGHT = 495;
        int[][] pixels= new int[WIDTH][HEIGHT];
        pixels = readImage("ass3-labeled.pgm", WIDTH, HEIGHT);
        Building[] buildingList = new Building[28]; // 0-27, but will only count 1-27
        HashMap<Integer, ArrayList<String>> buildingDescriptions = new HashMap<>();

        //Initializing the buildings
        for(int a = 0; a<28; a++){
                Building b = new Building(a, 0);
                buildingList[a] = b;
        }

        //Setting the areas
        int a = 0;
        for(int i = 0; i < WIDTH; i++){
                for(int j = 0; j < HEIGHT; j++){
                        int pixelVal = pixels[i][j];
                        //if(!((i==0) && ((j==0) || (j==1) || (j==2) || (j==3)))){
                                //System.out.println("pixelVal " + pixelVal + " " + i + " "+ j);
                                //Increment the area for each pixel belonging to that building
                                buildingList[pixelVal].setArea(buildingList[pixelVal].getArea() +1);
                        //}
                        if(pixelVal==17){
                                //System.out.println(" 255555555: " + i + " " + j);
                                a++;
                        }
                }
        }

        smallest = 10000;
        largest = 0;
        //Adding the area, and whether the building is small, medium or large to the descriptions
        for(int c = 1; c < 28; c++){
                //System.out.println("area: " + buildingList[c].getArea());
                ArrayList<String> descr = new ArrayList<>();
                String areaSentence = "It has area " + Integer.toString(buildingList[c].getArea());
                descr.add(areaSentence);
                if(isSmall(buildingList[c])){
                        String smallSentence = "It is a small building";
                        descr.add(smallSentence);
                }
                if(isMedium(buildingList[c])){
                        String mediumSentence = "It is a medium building";
                        descr.add(mediumSentence);
                }
                if(isLarge(buildingList[c])){
                        String largeSentence = "It is a large building";
                        descr.add(largeSentence);
                }
                if(buildingList[c].getArea() < smallest){
                        smallest = buildingList[c].getArea();
                        magic_smallest = c; //the number of the smallest building
                }
                if(buildingList[c].getArea() > largest){
                        largest = buildingList[c].getArea();
                        magic_largest = c; //the number of the largest building
                }
                buildingDescriptions.put(c, descr);
        }

        //Adding smallest and largest description
        ArrayList<String> tempDescr = buildingDescriptions.get(magic_smallest);
        String smallestSentence = "It is the smallest building";
        tempDescr.add(smallestSentence);
        buildingDescriptions.put(magic_smallest, tempDescr);

        ArrayList<String> tempLargeDescr = buildingDescriptions.get(magic_largest);
        String largestSentence = "It is the largest building";
        tempLargeDescr.add(largestSentence);
        buildingDescriptions.put(magic_largest, tempLargeDescr);


        //Adding the building names
        try (BufferedReader br = new BufferedReader(new FileReader("ass3-table.txt"))) {
          String line;
          while ((line = br.readLine()) != null) {
                String delims = "=";
                String[] tokens = line.split(delims);
                //System.out.println("tokens: " + tokens[1]);
                buildingList[Integer.parseInt(tokens[0])].setBuildingName(tokens[1]);
                ArrayList<String> nameDescr = buildingDescriptions.get(Integer.parseInt(tokens[0]));
                String nameSentence = "Its name is " + tokens[1];
                //System.out.println("name " + nameSentence);
                //nameDescr.add(nameSentence);
                //buildingDescriptions.put(Integer.parseInt(tokens[0]), nameDescr); //adding name to
descr
          }
```

```
} catch (IOException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
}

//Adding the moments and center of mass
ArrayList<int[]> buildingMoments = determineCenterOfMass("ass3-campus.jpg");
for(int e = 0; e < buildingMoments.size(); e++){
        int buildNum = pixels[buildingMoments.get(e)[0]][buildingMoments.get(e)[1]];
        Building b = buildingList[buildNum];
        //System.out.println("buildingNum " + buildNum);
        b.setCenterOfMassX(buildingMoments.get(e)[0]);
        b.setCenterOfMassY(buildingMoments.get(e)[1]);
        //System.out.println("x " + b.getCenterOfMassX());
        //System.out.println("y " + b.getCenterOfMassY());
}

//Add Border Sentences and NorthernMost/etc
Mat image = getMat("ass3-campus.jpg");
for(int e = 1; e < 28; e++){
        ArrayList<String> borderDescr = buildingDescriptions.get(e);
        if(isLocatedOnBorder(pixels, buildingList[e], WIDTH, HEIGHT)){
                String borderSentence = "It is located on the border";
                borderDescr.add(borderSentence);
        }else{
                String borderSentence = "It is located centrally";
                borderDescr.add(borderSentence);
        }
        if(isNorthernMost(buildingList) == e){
                String northernMostSentence = "It is the northern most building.";
                borderDescr.add(northernMostSentence);
        }else if(isSouthernMost(buildingList) == e){
                String southernMostSentence = "It is the souther most building";
                borderDescr.add(southernMostSentence);
        }else if(isEasternMost(buildingList) == e){
                String easternMostSentence = "It is the eastern most building.";
                borderDescr.add(easternMostSentence);
        }else if(isWesternMost(buildingList) == e){
                String westernMostSentence = "It is the western most building";
                borderDescr.add(westernMostSentence);
        }
        if(isRectangle(buildingList[e], image)){
                String recSentence = "It is a rectangle";
                borderDescr.add(recSentence);
        }else if(isSquare(buildingList[e], image)){
                String sqSentence = "It is a sqaure";
                borderDescr.add(sqSentence);
        }else{
                String sent = "It is not a rectangle or a square";
                borderDescr.add(sent);
                }
                if(isIShaped(buildingList[e], image)){
                        String sent = "It is I-Shaped";
                        borderDescr.add(sent);
                }else if(isLShaped(buildingList[e], image)){
                        String sent = "It is L-Shaped";
                        borderDescr.add(sent);
                }
                buildingDescriptions.put(e, borderDescr);
        }


        //Test
        for(int d = 1; d < 28; d++){
                Building b = buildingList[d];
                ArrayList<String> descr = buildingDescriptions.get(d);
                System.out.println("Building Number: " + b.getBuildingNumber());
                System.out.println("Building Name: " + b.getBuildingName());
                System.out.println("Center of Mass Coors: " + b.getCenterOfMassX() + " ," +
b.getCenterOfMassY());
                for(String s: descr){
                        System.out.println(s);
                }
                System.out.println("");
        }


        Core.circle(image, new Point(38, 441), 4, new Scalar(255,40,0,255));
        Highgui.imwrite("test.jpg",image);
        //System.out.println("pixval: " + pixels[0][0]); //142 35

        globBuildings = buildingList;

    }//end run

    public Building[] getBuildings(){
        return globBuildings;
    }


    public static int[][] readImage(String fileName, int newWidth, int newHeight)
    {
        int WIDTH = newWidth;
        int HEIGHT = newHeight;
        int[][] pixels = new int[WIDTH][HEIGHT];
        String line;
        StringTokenizer st;

        try {
      BufferedReader in =
      new BufferedReader(new InputStreamReader(
        new BufferedInputStream(
          new FileInputStream(fileName))));

    DataInputStream in2 =
      new DataInputStream(
        new BufferedInputStream(
          new FileInputStream(fileName)));

    // read PPM image header

    // skip comments
    line = in.readLine();
    //System.out.println("line: " + line);
    in2.skip((line+"\n").getBytes().length);
    do {
       line = in.readLine();
       in2.skip((line+"\n").getBytes().length);
       //System.out.println("saw #");
    } while (line.charAt(0) == '#');

    //read pixels now
    int a = 0;
    int b = 0;
    int pix0 = in2.readUnsignedByte();//header
    int pix1 = in2.readUnsignedByte();
    int pix2 = in2.readUnsignedByte();
    int pix3 = in2.readUnsignedByte();
    for (int c = 0; c < WIDTH; c++){
       for (int r = 0; r < HEIGHT; r++){
                    //int x = in2.readUnsignedByte();
                    //System.out.println("x : " + x);
                    int pix = in2.readUnsignedByte();
                    pixels[c][r] = pix;
                    a++;
                    if(a<19){
                                //System.out.println("pix: " + pix);
                    }
          }
          b++;
    }//outer for
    //System.out.println("a: " + a);
    //System.out.println("b: " + b);
    in.close();
    in2.close();
  } catch(ArrayIndexOutOfBoundsException e) {
    System.out.println("Error: image in "+fileName+" too big");
  } catch(FileNotFoundException e) {
    System.out.println("Error: file "+fileName+" not found");
  } catch(IOException e) {
```

```
                System.out.println("Error: end of stream encountered when reading "+fileName);
            }
                return pixels;
        }

        public ArrayList<int[]> determineCenterOfMass(String filename){
                Mat hu = new Mat();

                // Find the contours
                Mat image = getMat(filename);
            Mat imageHSV = new Mat(image.size(), Core.DEPTH_MASK_8U);
            Mat imageBlurr = new Mat(image.size(), Core.DEPTH_MASK_8U);
            Mat imageThresh = new Mat(image.size(), Core.DEPTH_MASK_ALL);
            Mat imageCanny = new Mat(image.size(), Core.DEPTH_MASK_ALL);

            Imgproc.cvtColor(image, imageHSV, Imgproc.COLOR_BGR2GRAY);
            //Imgproc.GaussianBlur(imageHSV, imageBlurr, new Size(5,5), 0);
            Imgproc.adaptiveThreshold(imageHSV, imageThresh, 255,Imgproc.ADAPTIVE_THRESH_MEAN_C,
Imgproc.THRESH_BINARY,7, 5);
            Imgproc.Canny(image, imageCanny, 100, 200);
            //Imgproc.Canny(imageHSV, imageCanny, 100, 200);
            Highgui.imwrite("Edges.jpg",imageCanny);

            List<MatOfPoint> contours = new ArrayList<MatOfPoint>();
            Imgproc.findContours(imageCanny, contours, new Mat(),
Imgproc.RETR_LIST,Imgproc.CHAIN_APPROX_SIMPLE);
            globContours = contours;
            System.out.println("Contour size: " + contours.size());

            // Draw the contours
            //Mat drawing = new Mat(image.size(), Core.DEPTH_MASK_8U);
            Mat mask = Mat.zeros(image.rows(),image.cols(),image.type());
            for( int i = 0; i< contours.size(); i++ )
            {
              //Scalar color = new Scalar( 0,0,255);
              //Imgproc.drawContours(drawing, contours, i, color, 1);
              Imgproc.drawContours(mask, contours, -1, new Scalar(0,0,255));
              //System.out.println("contourArea: " + Imgproc.contourArea(contours.get(i)));

            }
            Highgui.imwrite("Contours.jpg",mask);

            //Find the moments
            ArrayList<int[]> buildingMoments = new ArrayList<int[]>();
            System.out.println("contour size: " + contours.size());
            List<Moments> mu = new ArrayList<Moments>(contours.size());
            for (int i = 0; i < contours.size(); i++) {
              mu.add(i, Imgproc.moments(contours.get(i), false));
              Moments p = mu.get(i);
              int x = (int) (p.get_m10() / p.get_m00());
        int y = (int) (p.get_m01() / p.get_m00());
        int[] moms = new int[2];
        moms[0] = x;
        moms[1] = y;
        //if(((x==0) && (y==0)) || ((x==38) && (y==458)) || ((x==38) && (y==457))){
        if(((x==0) && (y==0))){
            //System.out.println("I has!");
            continue;
        }else if(doesContain(buildingMoments, moms)){
            //System.out.println("Has more!");
            continue;
        }
        else{
            buildingMoments.add(moms);
            //System.out.println("moms: " + i + " x: " + moms[0] + " y: " + moms[1]);
        }
        //System.out.println("moments: " + i + " x: "+ x + " y: "+ y);
        Core.circle(image, new Point(x, y), 4, new Scalar(255,40,0,255));
    }
    Highgui.imwrite("Moments.jpg",image);

    return buildingMoments;
}

public boolean doesContain(ArrayList<int[]> list, int[] value){
        boolean contains = false;
        for(int i = 0; i < list.size(); i++){
                if((list.get(i)[0] == value[0]) && (list.get(i)[1] == value[1])){
                        contains = true;
                }
        }
        return contains;
}

public Mat getMat(String filename){
        //System.out.println("fileeeeeeeeeeeeee: " + filename);
        Mat image = Highgui.imread(getClass().getResource(filename).getPath());
        return image;
}

public boolean isSmall(Building s){
        boolean isSmallBool = false;
        int area = s.getArea(); //TODO get area from pixels
        if(area<magic_small){
                isSmallBool= true;
        }
        return isSmallBool;
}

public boolean isMedium(Building s){
		boolean isMediumBool = false;
		int area = s.getArea(); //TODO get area from pixels
		if(area<magic_large && area>magic_small){
			isMediumBool= true;
		}
		return isMediumBool;
	}

	public boolean isLarge(Building s){
		boolean isLargeBool = false;
		int area = s.getArea(); //TODO get area from pixels
		if(area>magic_large){
			isLargeBool= true;
		}
		return isLargeBool;
	}

	public boolean isRectangle(Building s, Mat image){
		boolean isRectangle = true;
		int bNum = s.getBuildingNumber();
		Mat subimage  = image.submat(s.getCenterOfMassY()-s.getHeight()/2,
s.getCenterOfMassY()+s.getHeight()/2, s.getCenterOfMassX()-s.getWidth()/2,
s.getCenterOfMassX()-s.getWidth()/2);
		Mat template = Highgui.imread(getClass().getResource("rectangle.jpg").getPath());
		int result_cols = subimage.cols() - template.cols() + 1;
		int result_rows = subimage.rows() - template.rows() + 1;
		Mat result = new Mat(result_rows, result_cols, CvType.CV_32FC1);
		Imgproc.matchTemplate(subimage, subimage, result, Imgproc.TM_CCOEFF_NORMED);

		for (int i = 0; i < result_rows; i++){
			for (int j = 0; j < result_cols; j++) {
				if(result.get(i, j)[0]>0){
				 isRectangle = true;
				}
			}
		}
		return isRectangle;
	}

	public boolean isSquare(Building s, Mat image){
		boolean isSquare = false;
		int bNum = s.getBuildingNumber();
		Mat subimage  = image.submat(s.getCenterOfMassY()-s.getHeight()/2,
s.getCenterOfMassY()+s.getHeight()/2, s.getCenterOfMassX()-s.getWidth()/2,
s.getCenterOfMassX()-s.getWidth()/2);
		Mat template = Highgui.imread(getClass().getResource("square.jpg").getPath());
		int result_cols = subimage.cols() - template.cols() + 1;
		int result_rows = subimage.rows() - template.rows() + 1;
		Mat result = new Mat(result_rows, result_cols, CvType.CV_32FC1);
		Imgproc.matchTemplate(subimage, subimage, result, Imgproc.TM_CCOEFF_NORMED);
```

```
			for (int i = 0; i < result_rows; i++){
				for (int j = 0; j < result_cols; j++) {
					if(result.get(i, j)[0]>0){
					 isSquare = true;
					}
				}
			}

			return isSquare;
		}

		public boolean isNonRectangle(Building s, Mat image){
			boolean isNonRectangle = false;
			if(!isSquare(s, image) && !isRectangle(s, image)){
				isNonRectangle = true;
			}
			return isNonRectangle;
		}

		public boolean isIShaped(Building s, Mat image){
			boolean isIShaped = false;
			int bNum = s.getBuildingNumber();
			Mat subimage  = image.submat(s.getCenterOfMassY()-s.getHeight()/2,
s.getCenterOfMassY()+s.getHeight()/2, s.getCenterOfMassX()-s.getWidth()/2,
s.getCenterOfMassX()-s.getWidth()/2);
			Mat template = Highgui.imread(getClass().getResource("iShape.jpg").getPath());
			int result_cols = subimage.cols() - template.cols() + 1;
			int result_rows = subimage.rows() - template.rows() + 1;
			Mat result = new Mat(result_rows, result_cols, CvType.CV_32FC1);
			Imgproc.matchTemplate(subimage, subimage, result, Imgproc.TM_CCOEFF_NORMED);

			for (int i = 0; i < result_rows; i++){
				for (int j = 0; j < result_cols; j++) {
					if(result.get(i, j)[0]>0){
					 isIShaped = true;
					}
				}
			}
			return isIShaped;
		}

		public boolean isLShaped(Building s, Mat image){
			boolean isLShaped = false;
			int bNum = s.getBuildingNumber();
			Mat subimage  = image.submat(s.getCenterOfMassY()-s.getHeight()/2,
s.getCenterOfMassY()+s.getHeight()/2, s.getCenterOfMassX()-s.getWidth()/2,
s.getCenterOfMassX()-s.getWidth()/2);
			Mat template = Highgui.imread(getClass().getResource("lShape.jpg").getPath());
			int result_cols = subimage.cols() - template.cols() + 1;
```

```
        int result_rows = subimage.rows() - template.rows() + 1;
        Mat result = new Mat(result_rows, result_cols, CvType.CV_32FC1);
        Imgproc.matchTemplate(subimage, subimage, result, Imgproc.TM_CCOEFF_NORMED);

        for (int i = 0; i < result_rows; i++){
                for (int j = 0; j < result_cols; j++) {
                        if(result.get(i, j)[0]>0){
                         isLShaped = true;
                        }
                }
        }
        return isLShaped;
}

public boolean isLocatedCentrally(){
        boolean isLocatedCentrally = false;
        //Opposite of isLocatedOnBorder()
        return isLocatedCentrally;
}

public boolean isLocatedOnBorder(int[][] pixels, Building b, int WIDTH, int HEIGHT){
        boolean isLocatedOnBorder = false;
        //north
        for(int i = 0; i < WIDTH; i++){
                if(pixels[i][3] == b.getBuildingNumber()){
                        isLocatedOnBorder = true;
                }
        }
        //south
        for(int i = 0; i < WIDTH; i++){
                if(pixels[i][490] == b.getBuildingNumber()){
                        isLocatedOnBorder = true;
                }
        }
        //west
        for(int i = 0; i < HEIGHT; i++){
                if(pixels[4][i] == b.getBuildingNumber()){
                        isLocatedOnBorder = true;
                }
        }
        //east
        for(int i = 0; i < HEIGHT; i++){
                if(pixels[270][i] == b.getBuildingNumber()){
                        isLocatedOnBorder = true;
                }
        }
        return isLocatedOnBorder;
}

public int isNorthernMost(Building[] newBuildings){
		int isNorthernMost = 0;
		int northernVal = 495;
		Building[] buildings = newBuildings;
		for(int i = 1; i < buildings.length; i++){
			if(buildings[i].getCenterOfMassY() < northernVal){
				isNorthernMost = i;
				northernVal = buildings[i].getCenterOfMassY();
			}
		}
		return isNorthernMost;
	}

	public int isSouthernMost(Building[] newBuildings){
		int isSouthernMost = 0;
		int southernVal = 0;
		Building[] buildings = newBuildings;
		for(int i = 1; i < buildings.length; i++){
			if(buildings[i].getCenterOfMassY() > southernVal){
				isSouthernMost = i;
				southernVal = buildings[i].getCenterOfMassY();
			}
		}
		return isSouthernMost;
	}

	public int isEasternMost(Building[] newBuildings){
		int isEasternMost = 0;
		int easternVal = 0;
		Building[] buildings = newBuildings;
		for(int i = 1; i < buildings.length; i++){
			if(buildings[i].getCenterOfMassX() > easternVal){
				isEasternMost = i;
				easternVal = buildings[i].getCenterOfMassX();
			}
		}
		return isEasternMost;
	}

	public int isWesternMost(Building[] newBuildings){
		int isWesternMost = 0;
		int westernVal = 275;
		Building[] buildings = newBuildings;
		for(int i = 1; i < buildings.length; i++){
			if(buildings[i].getCenterOfMassX() < westernVal){
				isWesternMost = i;
				westernVal = buildings[i].getCenterOfMassX();
			}
		}
		return isWesternMost;
	}
```

```
        public static void main( String[] args )
        {
                System.loadLibrary( Core.NATIVE_LIBRARY_NAME );
                WhatDescriptions whatDescriptions = new WhatDescriptions();
                whatDescriptions.run();
        }
}
```

```
import java.awt.Graphics;
import java.awt.Image;
import java.awt.MouseInfo;
import java.awt.Point;
import java.awt.PointerInfo;
import java.awt.image.ImageObserver;
import java.io.File;
import java.io.IOException;
import java.util.ArrayList;
import java.util.HashMap;

import javax.imageio.ImageIO;

import org.opencv.core.Core;


public class WhereDescriptions {

        public void run(Building[] buildings){
                HashMap<Integer, ArrayList<int[]>> buildingDescriptions = new HashMap<>();

                Image image = null;
                try{
      File image2 = new File("ass3-campus.jpg");
      image = ImageIO.read(image2);

    }
    catch (IOException e){
      e.printStackTrace();
    }
                int imageWidth = image.getWidth((ImageObserver) this);
    int imageHeight = image.getHeight((ImageObserver) this);

    Graphics g = null;
    g.drawImage(image, 50, 50, (ImageObserver) this);

                Building b1 = new Building(28, 1);
                PointerInfo a0 = MouseInfo.getPointerInfo();
```

```
Point b = a0.getLocation();
int x = (int) b.getX();
int y = (int) b.getY();
b1.setCenterOfMassX(x);
b1.setCenterOfMassY(y);
Building b2 = new Building(29, 1);
PointerInfo a1 = MouseInfo.getPointerInfo();
Point b11 = a1.getLocation();
int x1 = (int) b11.getX();
int y1 = (int) b11.getY();
b2.setCenterOfMassX(x1);
b2.setCenterOfMassY(y1);
Building b3 = new Building(30, 1);
PointerInfo a2 = MouseInfo.getPointerInfo();
Point b22 = a2.getLocation();
int x2 = (int) b22.getX();
int y2 = (int) b22.getY();
b3.setCenterOfMassX(x2);
b3.setCenterOfMassY(y2);
Building b4 = new Building(31, 1);
PointerInfo a3 = MouseInfo.getPointerInfo();
Point b33 = a3.getLocation();
int x3 = (int) b33.getX();
int y3 = (int) b33.getY();
b4.setCenterOfMassX(x3);
b4.setCenterOfMassY(y3);
Building b5 = new Building(32, 1);
PointerInfo a4 = MouseInfo.getPointerInfo();
Point b44 = a4.getLocation();
int x4 = (int) b44.getX();
int y4 = (int) b44.getY();
b5.setCenterOfMassX(x4);
b5.setCenterOfMassY(y4);
Building b6 = new Building(33, 1);
PointerInfo a5 = MouseInfo.getPointerInfo();
Point b55 = a5.getLocation();
int x5 = (int) b55.getX();
int y5 = (int) b55.getY();
b6.setCenterOfMassX(x5);
b6.setCenterOfMassY(y5);
//buildingDescriptions = compareBuildings(buildingDescriptions, buildings);
buildingDescriptions = compareOneBuilding(buildingDescriptions, buildings, b6);
buildingDescriptions = reduceDescriptions(buildingDescriptions, buildings, b6);

//Test
/*for(int d = 1; d < 28; d++){
        Building b = buildings[d];
        ArrayList<int[]> descr = buildingDescriptions.get(d);
        //System.out.println("Building Number: " + b.getBuildingNumber());
        for(int a = 0; a < descr.size(); a++){
```

```
						if(descr.get(a)[0]==0){
							System.out.println("Building " + descr.get(a)[1] + " is near Building" +
descr.get(a)[2]);
						}else if(descr.get(a)[0]==1){
							System.out.println("Building " + descr.get(a)[1] + " is north of Building"
+ descr.get(a)[2]);
						}else if(descr.get(a)[0]==2){
							System.out.println("Building " + descr.get(a)[1] + " is south of Building"
+ descr.get(a)[2]);
						}else if(descr.get(a)[0]==3){
							System.out.println("Building " + descr.get(a)[1] + " is east of Building" +
descr.get(a)[2]);
						}else if(descr.get(a)[0]==1){
							System.out.println("Building " + descr.get(a)[1] + " is west of Building"
+ descr.get(a)[2]);
						}
					}
					System.out.println("");
			}*/

			//Test One
					ArrayList<int[]> descr = buildingDescriptions.get(b1.getBuildingNumber());
					//System.out.println("Building Number: " + b.getBuildingNumber());
					System.out.println("size " + descr.size());
					for(int a = 0; a < descr.size(); a++){
						if(descr.get(a)[0]==0){
							System.out.println("Building " + descr.get(a)[1] + " is near Building " +
descr.get(a)[2]);
						}else if(descr.get(a)[0]==1){
							System.out.println("Building " + descr.get(a)[1] + " is north of Building
" + descr.get(a)[2]);
						}else if(descr.get(a)[0]==2){
							System.out.println("Building " + descr.get(a)[1] + " is south of Building
" + descr.get(a)[2]);
						}else if(descr.get(a)[0]==3){
							System.out.println("Building " + descr.get(a)[1] + " is east of Building "
+ descr.get(a)[2]);
						}else if(descr.get(a)[0]==1){
							System.out.println("Building " + descr.get(a)[1] + " is west of Building "
+ descr.get(a)[2]);
						}
					}

		}

		public HashMap<Integer, ArrayList<int[]>> compareOneBuilding(HashMap<Integer, ArrayList<int[]>>
descriptions, Building[] buildings, Building b){
			HashMap<Integer, ArrayList<int[]>> buildingDescriptions = descriptions;

					ArrayList<int[]> descr = new ArrayList<int[]>();
```

```
            for(int j = 1; j < buildings.length; j++){

                    if(isNear(b, buildings[j])){
                        //System.out.println("Building " + b.getBuildingNumber() + "
is near Building " + buildings[j].getBuildingNumber());
                        int[] near = {0, b.getBuildingNumber(),
buildings[j].getBuildingNumber()};
                        descr.add(near);
                    }
                    if(isNorth(b, buildings[j])){
                        int[] north = {1, b.getBuildingNumber(),
buildings[j].getBuildingNumber()};
                        //System.out.println("Building " + b.getBuildingNumber() + "
is north of Building " + buildings[j].getBuildingNumber());
                        descr.add(north);
                    }
                    if(isSouth(b, buildings[j])){
                        int[] south = {2, b.getBuildingNumber(),
buildings[j].getBuildingNumber()};
                        System.out.println("Building " + b.getBuildingNumber() + " is
south of Building " + buildings[j].getBuildingNumber());
                        descr.add(south);
                    }
                    if(isEast(b, buildings[j])){
                        int[] east = { 3, b.getBuildingNumber(),
buildings[j].getBuildingNumber()};
                        System.out.println("Building " + b.getBuildingNumber() + " is
east of Building " + buildings[j].getBuildingNumber());
                        descr.add(east);
                    }
                    if(isWest(b, buildings[j])){
                        int[] west = { 4, b.getBuildingNumber(),
buildings[j].getBuildingNumber()};
                        System.out.println("Building " + b.getBuildingNumber() + " is
west of Building " + buildings[j].getBuildingNumber());
                        descr.add(west);
                    }
            }
            System.out.println("des size " + descr.size());
            buildingDescriptions.put(b.getBuildingNumber(), descr);
            System.out.println("");

        return buildingDescriptions;
    }

    public HashMap<Integer, ArrayList<int[]>> compareBuildings(HashMap<Integer, ArrayList<int[]>>
descriptions, Building[] buildings){
        HashMap<Integer, ArrayList<int[]>> buildingDescriptions = descriptions;
        for(int i = 1; i < buildings.length; i++){
            ArrayList<int[]> descr = new ArrayList<int[]>();
```

```
                        for(int j = 1; j < buildings.length; j++){
                                if(i!=j){
                                        if(isNear(buildings[i], buildings[j])){
                                                //System.out.println("Building " +
buildings[i].getBuildingNumber() + " is near Building " + buildings[j].getBuildingNumber());
                                                int[] near = {0, buildings[i].getBuildingNumber(),
buildings[j].getBuildingNumber()};
                                                descr.add(near);
                                        }
                                        if(isNorth(buildings[i], buildings[j])){
                                                int[] north = {1, buildings[i].getBuildingNumber(),
buildings[j].getBuildingNumber()};
                                                //System.out.println("Building " +
buildings[i].getBuildingNumber() + " is north of Building " + buildings[j].getBuildingNumber());
                                                descr.add(north);
                                        }
                                        if(isSouth(buildings[i], buildings[j])){
                                                int[] south = {2, buildings[i].getBuildingNumber(),
buildings[j].getBuildingNumber()};
                                                descr.add(south);
                                        }
                                        if(isEast(buildings[i], buildings[j])){
                                                int[] east = { 3, buildings[i].getBuildingNumber(),
buildings[j].getBuildingNumber()};
                                                //System.out.println("Building " +
buildings[i].getBuildingNumber() + " is east of Building " + buildings[j].getBuildingNumber());
                                                descr.add(east);
                                        }
                                        if(isWest(buildings[i], buildings[j])){
                                                int[] west = { 4, buildings[i].getBuildingNumber(),
buildings[j].getBuildingNumber()};
                                                descr.add(west);
                                        }
                                }//end outer if
                        }
                        //System.out.println("des size " + descr.size());
                        buildingDescriptions.put(buildings[i].getBuildingNumber(), descr);
                        System.out.println("");
                }//end outer for
                return buildingDescriptions;
        }

        public HashMap<Integer, ArrayList<int[]>> reduceDescriptions(HashMap<Integer, ArrayList<int[]>>
descriptions, Building[] buildings, Building b1){
                HashMap<Integer, ArrayList<int[]>> buildingDescr = descriptions;
                //System.out.println("this size " + descriptions.size());
                for(int  i =0; i < descriptions.size(); i++){//each building
                        //ArrayList<int[]> descrs = buildingDescr.get(i); //for all compare
                        ArrayList<int[]> descrs = buildingDescr.get(b1.getBuildingNumber()); //for one compare
                        //System.out.println("that size " + descrs.size());
                //ArrayList<int[]> tempDescrs = new ArrayList<int[]>();
                //tempDescrs = descrs;
                ArrayList<int[]> tempDescrs = new ArrayList<int[]>(descrs);
                //System.out.println("size " + descrs.size());
                for(int a = 0; a < descrs.size(); a++){//each description
                    //System.out.println("size a " + i);
                    for(int b= 0; b < descrs.size(); b++){
                        int[] tempB = descrs.get(b);
                        //System.out.println("TempB " + b);
                        //System.out.println("sizeee: " + descrs.size());
                        //System.out.println("TempA " + a);
                        int[] tempA = descrs.get(a);

                        if((a!=b) &&(tempA[0]==1) &&(tempB[0]==1)
&&(isNorth(buildings[tempA[2]], buildings[tempB[2]]))){
                            tempDescrs.remove(tempB);
                            System.out.println("removed north " + tempB[2]);
                        }
                        if((a!=b) &&(tempA[0]==2) &&(tempB[0]==2)
&&(isSouth(buildings[tempA[2]], buildings[tempB[2]]))){
                            tempDescrs.remove(tempB);
                            System.out.println("removed south " + tempB[2]);
                        }
                        if((a!=b) &&(tempA[0]==3) &&(tempB[0]==3)
&&(isEast(buildings[tempA[2]], buildings[tempB[2]]))){
                            tempDescrs.remove(tempB);
                            System.out.println("removed esst " + tempB[2]);
                        }
                        if((a!=b) &&(tempA[0]==4) &&(tempB[0]==4)
&&(isWest(buildings[tempA[2]], buildings[tempB[2]]))){
                            tempDescrs.remove(tempB);
                            System.out.println("removed west " + tempB[2]);
                        }
                    }
                }
                buildingDescr.put(i, tempDescrs);
            }

            return buildingDescr;
        }

        public boolean isNorth(Building s,Building t){
            boolean isNorth = false;
            if(s.getCenterOfMassY() < t.getCenterOfMassY()){
                isNorth = true;
            }
            return isNorth;
        }

        public boolean isSouth(Building s,Building t){
```

```
		boolean isSouth = false;
		if(s.getCenterOfMassY() > t.getCenterOfMassY()){
			isSouth = true;
		}
		return isSouth;
	}

	public boolean isEast(Building s, Building t){
		boolean isEast = false;
		if(s.getCenterOfMassX() > t.getCenterOfMassX()){
			isEast = true;
		}
		return isEast;
	}

	public boolean isWest(Building s, Building t){
		boolean isWest = false;
		if(s.getCenterOfMassX() < t.getCenterOfMassX()){
			isWest = true;
		}
		return isWest;
	}

	public boolean isNear(Building s, Building t){
		boolean isNear = false;
		int x = Math.abs(s.getCenterOfMassX()-t.getCenterOfMassX());
		int y = Math.abs(s.getCenterOfMassY()-t.getCenterOfMassY());
		//System.out.println(t.getBuildingNumber());
		//System.out.println("xy " + x + ","+ y);
		if((x<75) && (y<100)){
			isNear = true;
			//System.out.println("true: " + x + ","+ y);
		}
		return isNear;
	}

	public static void main( String[] args )
	{
		 System.loadLibrary( Core.NATIVE_LIBRARY_NAME );
		 WhereDescriptions whereDescriptions = new WhereDescriptions();
		 WhatDescriptions what = new WhatDescriptions();
		 what.run();
		 Building[] buildings = what.getBuildings();
		 whereDescriptions.run(buildings);
	}
}
```