```java
import java.util.List;
import java.util.ArrayList;

public class Cluster{
  public final int id;
  public List pixels;
  public float meanR;
  public float meanG;
  public float meanB;
  public int num;

  public Cluster(int id){
    this.id = id;
    pixels = new ArrayList();
    meanR = 0;
    meanG = 0;
    meanB = 0;
  }

  public void addPixel(Pixel pixel){
    pixels.add(pixel);
    meanR = (meanR*num+pixel.data)/(num+1);
    meanG = (meanG*num+pixel.data)/(num+1);
    meanB = (meanB*num+pixel.data)/(num+1);
    num++;
  }

  public void removePixel(Pixel pixel){
    pixels.delete(pixel);
    meanR = (meanR*num-pixel.data)/(num-1);
    meanG = (meanG*num-pixel.data)/(num-1);
    meanB = (meanB*num-pixel.data)/(num-1);
    num--;
  }

  public int getMeanRGB(){
    return (((int)meanR)<<24)+(((int)meanG)<<16)+(((int)meanB)<<8);
  }

}
import java.util.List;
import java.util.ArrayList;
import java.io.*;
import java.awt.image.BufferedImage;

public class Image{
  public final static int RADIUS = 20;
  public final static int POWRADIUS = RADIUS*RADIUS;
  public final BufferedImage img;
  public final Pixel[] pixels;
  public final int width;
  public final int height;
  public final int length;

  public Image(String filename){
    BufferedImage tmp=null;
    try{
      tmp = ImageUtil.load(filename);
    }catch(IOException ex){
      System.err.println(ex);
      System.exit(1);
    }
    img = tmp;
    width = img.getWidth();
    height = img.getHeight();
    length = width*height;
    pixels = new Pixel[length];
  }

  public void initPixels(){
    for(int i=0;i<length;i++){
      int data = img.getRGB(i%width, i/width);
      pixels[i] = new Pixel(i%width, i/width, data, this);
    }
    for(int i=0;i<length;i++){
      int cx=i%width;
      int cy=i/width;
      ArrayList<Pixel> neighbors = new ArrayList<Pixel>();
      for(int y=-RADIUS;y<=RADIUS;y++){
        for(int x=-RADIUS;x<=RADIUS;x++){
          if( x*x+y*y <= POWRADIUS &&
              0 <= cx && cx< width && 0 <= cy && cy < height ){
            neighbors.add(pixels[cx+cy*width]);
          }
        }
      }
      pixels[i].setNeighbors(neighbors);

      ArrayList<Pixel> adjacents = new ArrayList<Pixel>();
      Pixel tmp;
      if((tmp = getUpper(pixels[i]))!=null){
        adjacents.add(tmp);
      }
      if((tmp = getLower(pixels[i]))!=null){
```

```java
        adjacents.add(tmp);
      }
      if((tmp = getLefter(pixels[i]))!=null){
        adjacents.add(tmp);
      }
      if((tmp = getRighter(pixels[i]))!=null){
        adjacents.add(tmp);
      }
      pixels[i].setAdjacents(adjacents);
    }
  }

  public Pixel getUpper(Pixel pixel){
    int pos = (pixel.y-1)*width+pixel.x;
    if(pos<0){
      return null;
    }
    return pixels[pos];
  }

  public Pixel getLower(Pixel pixel){
    int pos = (pixel.y+1)*width+pixel.x;
    if(length<=pos){
      return null;
    }
    return pixels[pos];
  }

  public Pixel getLefter(Pixel pixel){
    int x = pixel.x-1;
    if(x<0){
      return null;
    }
    return pixels[x+pixel.y*width];
  }

  public Pixel getRighter(Pixel pixel){
    int x = pixel.x+1;
    if(width <= x){
      return null;
    }
    return pixels[x+pixel.y*width];
  }

}
import java.awt.image.BufferedImage;
import java.awt.image.WritableRaster;
import java.io.*;
import javax.imageio.ImageIO;

public class ImageUtil{
  private ImageUtil(){
  }

  /**
     load a file and make an image data.
     @param filename
   */
  public static BufferedImage load(String filename) throws IOException{
    if (filename == null){
      return null;
    }
    File f = new File(filename);
    return ImageIO.read(f);
  }

  public static void save(BufferedImage image, String filename)
                                                        throws IOException{
    File file = new File(filename);
    ImageIO.write(image, getSuffix(filename), file);
  }

  private static String getSuffix(String fileName) {
    if (fileName == null)
      return null;
    int point = fileName.lastIndexOf(".");
    if (point != -1) {
      return fileName.substring(point+1);
    }
    return "png";
  }
}
import javafx.application.Application;
import javafx.application.Platform;
import javafx.stage.*;
import javafx.scene.*;
import javafx.geometry.*;
import javafx.scene.control.*;
import javafx.scene.layout.*;
import javafx.scene.shape.*;
import javafx.scene.canvas.*;
import javafx.scene.image.WritableImage;
import javafx.scene.input.MouseEvent;
import javafx.scene.paint.Color;
```

```java
import javafx.event.EventHandler;
import java.awt.image.BufferedImage;
import javafx.embed.swing.SwingFXUtils;

public class Main extends Application{
  Image image;
  GraphicsContext gc;
  WritableImage wi;

  public Main(){
    image = new Image("sample.bmp");
    image.initPixels();
    wi = new WritableImage(image.width, image.height);
  }

  public void start(Stage stage){
    Button init = new Button("original");
    init.setOnAction((ev)->{
        SwingFXUtils.toFXImage(image.img, wi);
        gc.drawImage(wi,0,0);
      }
      );
    Button step = new Button("step");
    step.setOnAction((ev)->{
        Platform.runLater(()->{



            /* show boarder lines on the original image */
            SwingFXUtils.toFXImage(image.img, wi);
            gc.drawImage(wi,0,0);
            for(Pixel pix : image.pixels){
              //if(pix.isBorder()){
              if(pix.x%10 == 0){
                gc.setStroke(Color.GREEN);
                gc.strokeRect(pix.x,pix.y,1,1);
              }
            }
          });
      }
      );
    Button finish = new Button("finish");
    finish.setOnAction((ev)->{

      }
      );
    Canvas can = new Canvas(image.width, image.height);
    HBox buttons = new HBox();
    VBox root = new VBox();
    root.getChildren().addAll(buttons, can);
    buttons.getChildren().addAll(init, step, finish);
    gc = can.getGraphicsContext2D();
    Scene scene = new Scene(root);
    stage.setScene(scene);
    stage.show();
  }

  public void init(){
  }

}
import java.util.List;
import java.util.ArrayList;

public class Pixel{
  public final int x, y;
  public final int data;
  public final Image image;
  public int id;
  private List<Pixel> neighbors;
  private List<Pixel> adjacents;

  public Pixel(int x, int y, int data, Image image){
    this.x = x;
    this.y = y;
    this.data = data;
    this.image = image;
  }

  public void setNeighbors(List<Pixel> pixels){
    neighbors = pixels;
  }

  public void setAdjacents(List<Pixel> pixels){
    adjacents = pixels;
  }

  public List<Pixel> getNeighbors(){
    return neighbors;
  }

  public List<Pixel> getAdjacents(){
    return adjacents;
  }
  public boolean isBorder(){
    for(Pixel tmp: adjacents){
      if(tmp.getID() != id){
        return true;
      }
    }
    return false;
  }

  public void setID(int id){
    this.id = id;
  }

  public int getID(){
    return id;
  }

}
```