Contents

1 Basic Test Results

```
1  ======================
2  ==== EX4 P2 TESTER ===
3  ======================
4
5
6  ===== EXTRACTING JAR =====
7
8
9  ===== CHECKING FILES =====
10
11
12 ===== ANALYZE README =====
13
14
15 ===== COMPILE CODE =====
16
17
18 ===== EXECUTE TESTS =====
19
20
21 ==== ERRORS ====
22 You passed all the presubmission tests :)
23
24
25 ==== DONE ====
```

2 README

shacharpollak
208781005

The method chooseChar uses charToBrightness method to get an array of char brightnesses.
charToBrightness does that by going over the array of chars and use getImg to get a boolean array of the charImg,
then count each "True" and divide the sum of whites with "16*16" (16 is the resolution).
Then chooseChars makes the linear stretch of that brightnesses array by using the method linearStretching
which uses the formula (that we were given) on brightness in that array.
Then chooseChars uses convertImageToAscii method by dividing the image to subImages in the desired sizes,
calculates each subImage average brightness with imageAverageBrightness method that goes over each pixel
and uses the formula we were givento cast each color to the suited grey shade,
then sums all greys and divides by 255(max of RGB).
After that, convertImageToAscii finds the char that its brightness is closest to the average for that subImage
and puts it in the suited location in the asciiArt (result array).
I chose not to sort the array of chars brightnesses because the sorting the array is nlogn and after that
finding for each sub-image it's closest char will be logn, so overall nlogn.
without sorting overall will be n^2 but n cannot go to infinity!, n is at maximum the number of valid ascii's
and so n is O(256) which means there is no difference (time-complexity wise) for us.

I chose to use a Hashset in the Shell class for the chars set. so that there are no duplicates,
plus fast insertion and removal.
I also used a HashSet in the second algorithms question,
so that there are no duplicate morse codes and insertion at O(1).

Algorithms:
find duplicates -
Explanation:
We set to variables, faster (numList[0]) and slower (numList[numList[0]]).
while they aren't equal we advance them (the faster 2x faster then the slower).
when they meet it means we are in a cycle.
then we set faster to 0 (so it starts from the start node), and slower is still set to the meet node (in the cycle).
now when we advance them both at the same speed they will meet at the node that it's index is the duplicate number.
that is becuase there are two cells that their value is that said index and that means they are duplicate.
Time Complexity:
we used Floyds cycle detection algorithms which is known to be O(n), then finding the duplicate is O(n)
as well because we go over maximum of n cells. so overall O(n) time.
space complexity:
we only used constant extra space so O(1) as needed.

uniqueMorseRepresentations -
Explanation:
I translate each word to its morse code and then insert to a hashset so there are no duplicates.
then I return the set size which is the correct answer.
Time Complexity:
making the array of letters in morse is (O(N)), the outer for loop goes over the words so it runs n times,
each time it goes over each letter of the word and adds to a string of current morse,
then insert to a set(which is O(1)), So each time takes O(length of word),
and overall it takes exactly O(S) (S is the sum of lengths of words).

```
                          (b"     7b;5K   K5     '<OKi  KbbKi    .bK   ibbbK!   bbbbb"
                          W@%    "@@i6@J j@6    (@@@@@6 #@#@@5   *@@"  1@@#@@H  @@###I
                          W@@J   %@@i'@% %@'   "@#I 'Kr ## .#@   B#@b  1@7 .$@* @B
                          W@##. /@%@i O@"@b    H@J      ##  #@  ;@K%#  1@7  .@# @B
                          W@r@2 ##7@i '@#@'    B@. .bbF #@b%@2  K@;(@/ 1@7   #@ @@@@O
                          W@ %@K@/(@i  5@O     %@; .B@# #@B@W   ##j+@$ 1@7   @# @#++"
                          W@ ;@@% (@i  *@*     *@6   B# ## B@i !@@@@@@.1@7  /@W @B
                          W@  5@" (@i  *@*      B@$<2@# ## !@# H@"  *@*1@&*5@@; @#***;
                          W@      (@i  *@*      .K#@@BJ ##  6@2@#   .@B1@@@@%"  @@@@@j
```

```
   rb/   Fbbbbjbbbbbb'bbbbb;(bb5J     'bbbKI    /55/   <<   .br.b(   +b' ;25j
   B@#   #@###K##@@##;@@###!$@##@%    !@@#@@K .B@@@@%. B@7  '@K;@$   W@;;@@@@2
  ;@B@"  #@      @#  '@%    $@; $@"   !@H .@# W@5..O@& B@@" '@K;@$   W@;b@" J' ";
  K@"@b  #@      @#  .@%    $@; 6@"   !@H ;@$.@#    #@.B#@#.'@K;@$   W@;O@5.   @#
  #B %#  #@BBH   @#  .@@@@2 $@HW@%    !@@@@#;!@&    W@"B#J@$'@K;@$   W@;;#@@$' 65
 i@WjH@/ #@WWK   @#  .@#++" $@B@#.    !@%j7@#"@$    %@;B# b@K@K'@$   W@;  j%@&
 H@@@@@W #@      @#  .@%    $@;5@2    !@H  %@.#@;  ;@# B#  B@@K @#   #@...  @B
 @#   B@.#@      @#  .@#***'$@;.#@;   !@%(F@# J@#b6@@j B#  '@@K b@BOB@6 $#K6@5 #%
+@2   <@*#@      @#  .@@@@@i$@; +@%   !@@@@B"  /B@@Br  B#   r@K  5#@@O  IB@@H. #B


                                        (.
                          'j.   '/;   "///!       .@K
                         "#@i  1@@@K  B@@@&        $#
                        r@@@i ;@%'W@! @6"".        J@I
                        J$2@i F@r "@O'@$5I     7@j .@2
                          1@i 6@;  @$!@##@1    (@/  @6
                          1@i 6@;  @$   .##         #&
                          1@i F@r "@O    %#         @b
                          1@i ;@%'W@!rb;r@$    !%" ;@<
                        .@@@@W 1@@@5 <@@@#'    K@< <@"
                         jj/j!  '/;   .r!      .I  B#
                                              .@J
                                               '
```

3 ascii art/Algorithms.java

```java
package ascii_art;
import java.util.HashSet;

public class Algorithms {
    /**
     * finds the duplicate in a list of number from 1 to n.
     * @param numList - list of numbers
     * @return - the duplicate
     */
    public static int findDuplicate(int[] numList){
        int faster = numList[numList[0]];
        int slower = numList[0];
        while(faster != slower){ // find the start of a cycle
            faster = numList[numList[faster]];
            slower = numList[slower];
        }
        faster = 0;
        while(faster != slower){ // find the duplicate in the cycle
            faster = numList[faster];
            slower = numList[slower];
        }
        return slower;
    }

    /**
     * calculates number of unique morse codes
     * @param words - list of strings
     * @return - the number of unique morse codes
     */
    public static int uniqueMorseRepresentations(String[] words){
        String[] arr = new String[]{".-","-...","-.-.","-..",".","..-.","--.","....","..",".---","-.-",
                ".-..","--","-.","---",".--.","--.-",".-.","...","-","..-","...-",".--","-..-","-.--","--.."};
        HashSet<String> set = new HashSet<>();
        for (String word : words){
            String cur = "";
            for (int i = 0; i < word.length(); i++) {
                cur = cur.concat(arr[word.toLowerCase().charAt(i) - 97]);
            }
            set.add(cur);
        }
        return set.size();
    }
}
```

4 ascii art/Driver.java

```java
package ascii_art;
import image.Image;
import java.util.logging.Logger;

/**
 * driver class for ascii art.
 */
public class Driver {
    /**
     * main function
     * @param args - arguments
     * @throws Exception - exception to throw
     */
    public static void main(String[] args) throws Exception {
        if (args.length != 1) {
            System.err.println("USAGE: java asciiArt ");
            return;
        }
        Image img = Image.fromFile(args[0]);
        if (img == null) {
            Logger.getGlobal().severe("Failed to open image file " + args[0]);
            return;
        }
        new Shell(img).run();
    }
}
```

5 ascii art/Shell.java

```java
package ascii_art;

import ascii_art.img_to_char.BrightnessImgCharMatcher;
import ascii_output.AsciiOutput;
import ascii_output.HtmlAsciiOutput;
import image.Image;

import java.util.*;
import java.util.stream.Stream;

/**
 * shell class of ascii art
 */
public class Shell {
    private static final String CMD_EXIT = "exit";
    private static final String SHOW_CHARS = "chars";
    private static final String ADD_CHARS = "add";
    private static final String REMOVE_CHARS = "remove";
    private static final String RES_CHANGE = "res";
    private static final String RENDER = "render";
    private static final String CONSOLE = "console";
    private static final int INITIAL_CHARS_IN_ROW = 64;
    private static final int MIN_PIXELS_PER_CHAR = 2;
    private static final String FONT_NAME = "Courier New";
    private static final String OUTPUT_FILENAME = "out.html";
    private static final String INITIAL_CHARS_RANGE = "0-9";
    private static final String RES_MAX_MSG = "Resolution is already at maximum value.";
    private static final String WIDTH_SET_TO = "Width set to: ";
    private static final String RES_UP = "up";
    private static final String RES_DOWN = "down";
    private static final String RES_MIN_MSG = "Resolution is already at minimum value.";
    private static final String RES_FORMAT = "ERROR: resolution change format is: res <up/down>";
    private static final String ERROR_INVALID_OPERATION = "ERROR: invalid operation";
    private static final String CHARS_FORMAT = "Show Chars command format is just: 'chars'";
    private static final String RENDER_FORMAT = "Render command format is just: 'render'";
    private static final String CONSOLE_FORMAT = "Console command format is just: 'console'";
    private static final String CONSOLE_GUI = ">>> ";
    private static final char FIRST_ASCII = ' ';
    private static final char LAST_ASCII = '~';
    private static final char SPACE = ' ';
    private static final char RANGE_SEPARATOR = '-';
    private static final int RES_FACTOR = 2;
    private static final int PARAM_LENGTH_ZERO = 0;
    private static final int PARAM_LENGTH_THREE = 3;


    private final Scanner scanner = new Scanner(System.in);
    private final Set<Character> charSet = new HashSet<>();

    private final int minCharsInRow;
    private final int maxCharsInRow;
    private int charsInRow;
    private final BrightnessImgCharMatcher charMatcher;
    private final AsciiOutput output;
    private boolean console = false;

    /**
     * constructor of shell class.
     * @param img - the image to work on.
```

```
60          */
61         public Shell(Image img) {
62             minCharsInRow = Math.max(1, img.getWidth()/img.getHeight());
63             maxCharsInRow = img.getWidth() / MIN_PIXELS_PER_CHAR;
64             charsInRow = Math.max(Math.min(INITIAL_CHARS_IN_ROW, maxCharsInRow), minCharsInRow);
65             charMatcher = new BrightnessImgCharMatcher(img, FONT_NAME);
66             output = new HtmlAsciiOutput(OUTPUT_FILENAME, FONT_NAME);
67             addChars(INITIAL_CHARS_RANGE);
68         }
69 
70         /**
71          * shows the current chars in the set.
72          */
73         private void showChars(){
74             charSet.stream().sorted().forEach(c-> System.out.print(c + " "));
75             System.out.println();
76         }
77 
78         /**
79          * parses the chars range
80          * @param param - the string to parse
81          * @return - the range
82          */
83         private static char[] parseCharRange(String param) {
84             if (param.length() == 1){
85                 return new char[]{param.charAt(0), param.charAt(0)};
86             }
87             else if (param.equals("all")){
88                 return new char[]{FIRST_ASCII, LAST_ASCII};
89             }
90             else if (param.equals("space")){
91                 return new char[]{SPACE, SPACE};
92             }
93             else if (param.length() == PARAM_LENGTH_THREE && param.charAt(1) == RANGE_SEPARATOR){
94                 if (param.charAt(0) > param.charAt(2)){ //opposite
95                     return new char[]{param.charAt(2), param.charAt(0)};
96                 }
97                 return new char[]{param.charAt(0), param.charAt(2)};
98             }
99             else{
100                 System.out.println(ERROR_INVALID_OPERATION);
101                 return null;
102             }
103         }
104 
105         /**
106          * adds chars to the set
107          * @param s - the string to parse
108          */
109         private void addChars(String s) {
110             char[] range = parseCharRange(s);
111             if(range != null){
112                 Stream.iterate(range[0], c -> c <= range[1], c -> (char)((int)c+1)).forEach(charSet::add);
113             }
114         }
115 
116         /**
117          * removes chars from the set
118          * @param s - the string to parse
119          */
120         private void removeChars(String s){
121             char[] range = parseCharRange(s);
122             if(range != null){
123                 Stream.iterate(range[0], c -> c <= range[1], c -> (char)((int)c+1)).forEach(charSet::remove);
124             }
125         }
126 
127         /**
```

```
128         * change the resolution
129         * @param s - the string to parse
130         */
131        private void resChange(String s){
132            if (s.equals(RES_UP)){
133                if (charsInRow * RES_FACTOR <= maxCharsInRow){
134                    charsInRow *= RES_FACTOR;
135                    System.out.println(WIDTH_SET_TO + charsInRow);
136                }
137                else{
138                    System.out.println(RES_MAX_MSG);
139                }
140            }
141            else if (s.equals(RES_DOWN)){
142                if (charsInRow/RES_FACTOR >= minCharsInRow){
143                    charsInRow /= RES_FACTOR;
144                    System.out.println(WIDTH_SET_TO + charsInRow);
145                }
146                else{
147                    System.out.println(RES_MIN_MSG);
148                }
149            }
150            else{
151                System.out.println(RES_FORMAT);
152            }
153        }
154
155        /**
156         * renders the output
157         */
158        private void render(){
159            Character[] arr = new Character[charSet.size()];
160            char[][] result = charMatcher.chooseChars(charsInRow, charSet.toArray(arr));
161            if (console){
162                for (char[] row : result) {
163                    for (char c : row) {
164                        System.out.print(c);
165                    }
166                    System.out.println();
167                }
168            }
169            else{
170                output.output(result);
171            }
172        }
173
174        /**
175         * runs the shell.
176         */
177        public void run() {
178            System.out.print(CONSOLE_GUI);
179            String cmd = scanner.next().trim();
180            while(!cmd.equals(CMD_EXIT)) {
181                var param = scanner.nextLine().trim();
182                switch (cmd) {
183                    case SHOW_CHARS:
184                        if (param.length() > PARAM_LENGTH_ZERO){
185                            System.out.println(CHARS_FORMAT);
186                            break;
187                        }
188                        showChars();
189                        break;
190                    case ADD_CHARS:
191                        addChars(param);
192                        break;
193                    case REMOVE_CHARS:
194                        removeChars(param);
195                        break;
```

```
                case RES_CHANGE:
                    resChange(param);
                    break;
                case RENDER:
                    if (param.length() > PARAM_LENGTH_ZERO){
                        System.out.println(RENDER_FORMAT);
                        break;
                    }
                    else if (charSet.isEmpty()){
                        break;
                    }
                    render();
                    break;
                case CONSOLE:
                    if (param.length() > PARAM_LENGTH_ZERO){
                        System.out.println(CONSOLE_FORMAT);
                        break;
                    }
                    console = true;
                    break;
                default:
                    System.out.println(ERROR_INVALID_OPERATION);
            }
            System.out.print(CONSOLE_GUI);
            cmd = scanner.next();
        }
    }
}

```

6 ascii art/packageinfo.java

```
1  /**
2   * Main module of the application.
3   * @author Dan Nirel
4   */
5  package ascii_art;
```

7 ascii art/img to char/BrightnessImgCharMatcher.java

```java
package ascii_art.img_to_char;

import image.Image;

import java.awt.*;
import java.util.HashMap;

/**
 * brightnessImgCharMatcher class for the AsciiArt.
 */
public class BrightnessImgCharMatcher {

    private static final int RGB_MAX = 255;
    private static final int RESOLUTION = 16;
    private static final double MULT_RED_FOR_GREY = 0.2126;
    private static final double MULT_GREEN_FOR_GREY = 0.7152;
    private static final double MULT_BLUE_FOR_GREY = 0.0722;
    public static final String ERROR_BAD_IMAGE = "ERROR: BAD IMAGE";
    private final Image img;
    private final String font;
    private final HashMap<Image, Float> cache = new HashMap<>();

    /**
     * constructor of the class.
     * @param img - an Image
     * @param font - a Font (string - name of font)
     */
    public BrightnessImgCharMatcher(Image img, String font) {

        this.img = img;
        this.font = font;
        if (img == null){
            System.out.println(ERROR_BAD_IMAGE);
        }
    }

    /**
     * chooses the chars for each subImage(size based on numCharsInRow) of the img
     * @param numCharsInRow - number of characters in a row
     * @param charSet - the set of characters to use
     * @return - Ascii art of the img.
     */
    public char[][] chooseChars(int numCharsInRow, Character[] charSet){
        float [] charToBrightness = charToBrightness(charSet);
        float[] charBrightnessLinearStretched = linearStretching(charToBrightness);
        char[][] result = convertImageToAscii(numCharsInRow, charSet, charBrightnessLinearStretched);
        return result;
    }

    /**
     * converts image to ascii.
     * @param numCharsInRow - number of characters in a row
     * @param charArr - the set of characters to use.
     * @param charBrightnessLinearStretched - character brightnesses after linear stretching.
     * @return - image converted to ascii.
     */
```

```
57      private char[][] convertImageToAscii(int numCharsInRow, Character[] charArr,
58                                           float[] charBrightnessLinearStretched){
59          int pixels = img.getWidth() / numCharsInRow;
60          char[][] asciiArt = new char[img.getHeight()/pixels][img.getWidth()/pixels];
61          if (charArr.length == 0){
62              return asciiArt;
63          }
64          int i = 0;
65          int j = 0;
66
67          for(Image subImage : img.squareSubImagesOfSize(pixels)) {
68              float imgAvgBrightness = imageAverageBrightness(subImage);
69              int idxBest = 0;
70              float bestDistance = 1;
71              for (int k = 0; k < charBrightnessLinearStretched.length; k++) {
72                  float distance = charBrightnessLinearStretched[k]-imgAvgBrightness;
73                  if (Math.abs(distance) < bestDistance){
74                      bestDistance = Math.abs(distance);
75                      idxBest = k;
76                  }
77              }
78              asciiArt[i][j] = charArr[idxBest];
79              j++;
80              if (j == numCharsInRow){
81                  j = 0;
82                  i++;
83              }
84          }
85          return asciiArt;
86      }
87
88      /**
89       * calculates image average brightness.
90       * @param img - Image
91       * @return - image average brightness.
92       */
93      private float imageAverageBrightness(Image img){
94          if (cache.containsKey(img)){
95              return cache.get(img);
96          }
97          float sum = 0;
98          int countPixels = 0;
99          for (Color pixel : img.pixels()) {
100             sum += (pixel.getRed() * MULT_RED_FOR_GREY +  pixel.getGreen() * MULT_GREEN_FOR_GREY +
101                     pixel.getBlue() * MULT_BLUE_FOR_GREY)/RGB_MAX;
102             countPixels++;
103         }
104         cache.put(img, sum/countPixels);
105         return sum/(countPixels);
106     }
107
108     /**
109      * linear stretching of the brightnesses in the array.
110      * @param brightnessArr - array of brightnesses
111      * @return - linear stretching of the brightnesses in the array.
112      */
113     private static float[] linearStretching(float[] brightnessArr){
114         float max = 0;
115         float min = 1;
116         for (float f: brightnessArr){
117             if (f < min){
118                 min = f;
119             }
120             if (f > max){
121                 max = f;
122             }
123         }
124         float[] stretched = new float[brightnessArr.length];
```

```
125             if (brightnessArr.length == 0 || min == max){
126                 return stretched;
127             }
128             for (int i = 0; i < stretched.length; i++) {
129                 stretched[i] = (brightnessArr[i]-min) /(max-min);
130             }
131             return stretched;
132         }
133 
134         /**
135          * Calculates each character's brightness.
136          * @param arr - array of characters
137          * @return - array of brightnesses
138          */
139         private float[] charToBrightness(Character[] arr){
140             float[] brightnessArr = new float[arr.length];
141             if (arr.length == 0){
142                 return brightnessArr;
143             }
144             for (int i = 0; i < arr.length; i++) {
145                 int whiteCount = 0;
146                 boolean[][] charBool = CharRenderer.getImg(arr[i], RESOLUTION, font);
147                 for(boolean[] row: charBool){
148                     for (boolean cell: row){
149                         if(cell){
150                             whiteCount++;
151                         }
152                     }
153                 }
154                 brightnessArr[i] = (float)whiteCount/(RESOLUTION*RESOLUTION);
155             }
156             return brightnessArr;
157         }
158     }
```

8 ascii art/img to char/CharRenderer.java

```java
package ascii_art.img_to_char;

import java.awt.*;
import java.awt.image.BufferedImage;

/**
 * Inspired by, and partly copied from
 * https://github.com/korhner/asciimg/blob/95c7764a6abe0e893fae56b3b6b580e09e1de209/src/main/java/io/korhner/asciimg/image/As
 * described in the blog:
 * https://dzone.com/articles/ascii-art-generator-java
 * Adaptations made by Dan Nirel.
 * The class renders (draws) characters to a binary "image" (2D array of booleans).
 */
public class CharRenderer {
    private static final double X_OFFSET_FACTOR = 0.2;
    private static final double Y_OFFSET_FACTOR = 0.75;

    /**
     * Renders a given character, according to how it looks in the font specified in the
     * constructor, to a square black&white image (2D array of booleans),
     * whose dimension in pixels is specified.
     */
    public static boolean[][] getImg(char c, int pixels, String fontName) {
        int key = (pixels << 8) | c;
        return render(c, pixels, fontName);
    }

    /**
     * renders the image
     * @param c - char
     * @param pixels - num of pixels
     * @param fontName - the font name
     * @return a boolean 2D array
     */
    private static boolean[][] render(char c, int pixels, String fontName) {
        String charStr = Character.toString(c);
        Font font = new Font(fontName, Font.PLAIN, pixels);
        BufferedImage img = new BufferedImage(pixels, pixels, BufferedImage.TYPE_INT_ARGB);
        Graphics g = img.getGraphics();
        g.setFont(font);
        int xOffset = (int)Math.round(pixels*X_OFFSET_FACTOR);
        int yOffset = (int)Math.round(pixels*Y_OFFSET_FACTOR);
        g.drawString(charStr, xOffset, yOffset);
        boolean[][] matrix = new boolean[pixels][pixels];
        for(int y = 0 ; y < pixels ; y++) {
            for(int x = 0 ; x < pixels ; x++) {
                matrix[y][x] = img.getRGB(x, y) == 0; //is the color black
            }
        }
        return matrix;
    }

    //for debugging
    /**
     * print the boolean array
     * @param arr - the array
     */
    public static void printBoolArr(boolean[][] arr) {
        for (boolean[] row : arr) {
```

```
60              for (boolean bool : row) {
61                  if (!bool) {
62                      System.out.print("#");
63                  } else {
64                      System.out.print(" ");
65                  }
66              }
67              System.out.println();
68          }
69      }
70  }
```

9 ascii art/img to char/packageinfo.java

```java
/**
 * The module responsible for actually translating images to chars
 * @author Dan Nirel
 */
package ascii_art.img_to_char;
```