```java
package edu.asu.msrs.artcelerationlibrary;

import android.content.Context;
import android.graphics.Bitmap;
import android.graphics.BitmapFactory;
import android.graphics.Region;
import android.util.Log;

import java.nio.ByteBuffer;

/**
 * Created by tangmiao on 11/27/2016.
 *The Ascii Art transform was basically use a huge amount of (9 x 17 x 4) ascii figuresto
cover the
 * original picture.  It was more replacing or changing a small area of pixelsthan transforming
 because
 * there was almost none calculation inside every pixel.  AsciiArt will cut the whole image
into a number
 * of small regions that each region has equalarea with the ascii images.Then the program will
 calculate
 * the average number foreach  region  and  use  an  ascii  image  with  the  closed  average
number  to
 * replace  theoriginal pixels inside it.
 */
public class AsciiArt {

    String TAG = "Ascii";

    byte[] char0;
    byte[] char1;
    byte[] char2;
    byte[] char3;
    byte[] char4;
    byte[] char5;
    byte[] char6;
    byte[] char7;
    byte[] char8;
    byte[] char9;
    byte[] char10;
    byte[] char11;
    byte[] char12;
    byte[] char13;
    byte[] char14;
    byte[] char15;
    byte[] char16;
    byte[] char17;
    byte[] char18;
    byte[] char19;
    byte[] char20;
    byte[] char21;
    byte[] char22;
    byte[] char23;
    byte[] char24;
    byte[] char25;
    byte[] char26;
    byte[] char27;
    byte[] char28;
    byte[] char29;
    byte[] char30;
    byte[] char31;
    byte[] char32;
    byte[] char33;
    byte[] char34;
    byte[] char35;

    int[] avg_ascii;

    int pixelswidth = 1600;

    byte[][] Asciimage;

    // Function: Converts bitmap object into byte array
    // input : bitmap format of image
    // output: Byte array of image
    public byte[] bmpToByte(Bitmap bitmap) {

        ByteBuffer buffer = ByteBuffer.allocateDirect(bitmap.getByteCount());
        bitmap.copyPixelsToBuffer(buffer);

        byte[] bytes = buffer.array();

        return bytes;
    }

    Context mContext;

    public AsciiArt(Context context) {

        mContext = context;
    }


    public byte[] ascii(byte[] pixels) {
        Log.d(TAG, "Start");


        // char0-7
        Bitmap imgbmp = BitmapFactory.decodeResource(mContext.getResources(), R.
drawable.char0);
        char0 = bmpToByte(imgbmp);
        Bitmap ch1bmp = BitmapFactory.decodeResource(mContext.getResources(), R.
drawable.char1);
        char1 = bmpToByte(ch1bmp);
        Bitmap ch2bmp = BitmapFactory.decodeResource(mContext.getResources(), R.
drawable.char2);
        char2 = bmpToByte(ch2bmp);
        Bitmap ch3bmp = BitmapFactory.decodeResource(mContext.getResources(), R.
drawable.char3);
        char3 = bmpToByte(ch3bmp);
        Bitmap ch4bmp = BitmapFactory.decodeResource(mContext.getResources(), R.
drawable.char4);
        char4 = bmpToByte(ch4bmp);
        Bitmap ch5bmp = BitmapFactory.decodeResource(mContext.getResources(), R.
drawable.char5);
        char5 = bmpToByte(ch5bmp);
        Bitmap ch6bmp = BitmapFactory.decodeResource(mContext.getResources(), R.
drawable.char6);
        char6 = bmpToByte(ch6bmp);
        Bitmap ch7bmp = BitmapFactory.decodeResource(mContext.getResources(), R.
drawable.char7);
        char7 = bmpToByte(ch7bmp);

        // char8-15
        Bitmap ch8bmp = BitmapFactory.decodeResource(mContext.getResources(), R.
drawable.char8);
        char8 = bmpToByte(ch8bmp);
        Bitmap ch9bmp = BitmapFactory.decodeResource(mContext.getResources(), R.
drawable.char9);
        char9 = bmpToByte(ch9bmp);
        Bitmap ch10bmp = BitmapFactory.decodeResource(mContext.getResources(), R
.drawable.char10);
        char10 = bmpToByte(ch10bmp);
        Bitmap ch11bmp = BitmapFactory.decodeResource(mContext.getResources(), R
.drawable.char11);
        char11 = bmpToByte(ch11bmp);
        Bitmap ch12bmp = BitmapFactory.decodeResource(mContext.getResources(), R
.drawable.char12);
        char12 = bmpToByte(ch12bmp);
        Bitmap ch13bmp = BitmapFactory.decodeResource(mContext.getResources(), R
.drawable.char13);
        char13 = bmpToByte(ch13bmp);
        Bitmap ch14bmp = BitmapFactory.decodeResource(mContext.getResources(), R
.drawable.char14);
        char14 = bmpToByte(ch14bmp);
        Bitmap ch15bmp = BitmapFactory.decodeResource(mContext.getResources(), R
.drawable.char15);
        char15 = bmpToByte(ch15bmp);
```

```java

        // char16-23
        Bitmap ch16bmp = BitmapFactory.decodeResource(mContext.getResources(), R
    .drawable.char16);
        char16 = bmpToByte(ch16bmp);
        Bitmap ch17bmp = BitmapFactory.decodeResource(mContext.getResources(), R
    .drawable.char17);
        char17 = bmpToByte(ch17bmp);
        Bitmap ch18bmp = BitmapFactory.decodeResource(mContext.getResources(), R
    .drawable.char18);
        char18 = bmpToByte(ch18bmp);
        Bitmap ch19bmp = BitmapFactory.decodeResource(mContext.getResources(), R
    .drawable.char19);
        char19 = bmpToByte(ch19bmp);
        Bitmap ch20bmp = BitmapFactory.decodeResource(mContext.getResources(), R
    .drawable.char20);
        char20 = bmpToByte(ch20bmp);
        Bitmap ch21bmp = BitmapFactory.decodeResource(mContext.getResources(), R
    .drawable.char21);
        char21 = bmpToByte(ch21bmp);
        Bitmap ch22bmp = BitmapFactory.decodeResource(mContext.getResources(), R
    .drawable.char22);
        char22 = bmpToByte(ch22bmp);
        Bitmap ch23bmp = BitmapFactory.decodeResource(mContext.getResources(), R
    .drawable.char23);
        char23 = bmpToByte(ch23bmp);

        // char24-31
        Bitmap ch24bmp = BitmapFactory.decodeResource(mContext.getResources(), R
    .drawable.char24);
        char24 = bmpToByte(ch24bmp);
        Bitmap ch25bmp = BitmapFactory.decodeResource(mContext.getResources(), R
    .drawable.char25);
        char25 = bmpToByte(ch25bmp);
        Bitmap ch26bmp = BitmapFactory.decodeResource(mContext.getResources(), R
    .drawable.char26);
        char26 = bmpToByte(ch26bmp);
        Bitmap ch27bmp = BitmapFactory.decodeResource(mContext.getResources(), R
    .drawable.char27);
        char27 = bmpToByte(ch27bmp);
        Bitmap ch28bmp = BitmapFactory.decodeResource(mContext.getResources(), R
    .drawable.char28);
        char28 = bmpToByte(ch28bmp);
        Bitmap ch29bmp = BitmapFactory.decodeResource(mContext.getResources(), R
    .drawable.char29);
        char29 = bmpToByte(ch29bmp);
        Bitmap ch30bmp = BitmapFactory.decodeResource(mContext.getResources(), R
    .drawable.char30);
        char30 = bmpToByte(ch30bmp);
        Bitmap ch31bmp = BitmapFactory.decodeResource(mContext.getResources(), R
.drawable.char31);
        char31 = bmpToByte(ch31bmp);
        Bitmap ch32bmp = BitmapFactory.decodeResource(mContext.getResources(), R
.drawable.char32);
        char32 = bmpToByte(ch32bmp);
        Bitmap ch33bmp = BitmapFactory.decodeResource(mContext.getResources(), R
.drawable.char33);
        char33 = bmpToByte(ch33bmp);
        Bitmap ch34bmp = BitmapFactory.decodeResource(mContext.getResources(), R
.drawable.char34);
        char34 = bmpToByte(ch34bmp);
        Bitmap ch35bmp = BitmapFactory.decodeResource(mContext.getResources(), R
.drawable.char35);
        char35 = bmpToByte(ch35bmp);

        ReqArgs mReq = new ReqArgs();
        //build an 2D Asciimage array to store all ascii images into a new array
        Asciimage = new byte[][]{char0, char1, char2, char3, char4, char5, char6, char7,
                char8, char9, char10,char11, char12, char13, char14, char15,
                char16, char17, char18, char19, char20, char21, char22, char23,
                char24, char25, char26, char27, char28, char29, char30, char31,char32,
char33,char34,char35};



        avg_ascii = new int[]{getAvg(char0), getAvg(char1), getAvg(char2), getAvg(
char3),
                getAvg(char4), getAvg(char5), getAvg(char6), getAvg(char7), getAvg(
char8), getAvg(char9),
                getAvg(char10), getAvg(char11), getAvg(char12), getAvg(char13), getAvg(
char14), getAvg(char15),
                getAvg(char16), getAvg(char17), getAvg(char18), getAvg(char19), getAvg(
char20), getAvg(char21),
                getAvg(char22), getAvg(char23), getAvg(char24), getAvg(char25), getAvg(
char26), getAvg(char27),
                getAvg(char28), getAvg(char29), getAvg(char30), getAvg(char31),getAvg(
char32), getAvg(char33), getAvg(char34),getAvg(char35)};



        int patchNumY = (int) Math.floor(1066/34);
        int patchNumX = (int) Math.floor(1600/18);
        int asciiImageIndex = 0;

        // i and j control the pixel insertion
        // k and p control the index of different regions on the original image

        byte[] outputImage= new byte[pixels.length];
```

```java
        for (int p = 0; p< patchNumY ; p ++) {  // W/w = 1066/34 , total patch in y direction
            for (int k = 0; k <patchNumX; k++) {  // total patch in x direction
                asciiImageIndex = findMin(k,p,pixels);

                for (int j = 0; j < 34; j++) {
                    for (int i = 0; i < 18 * 4; i++) {
                        int indexpixel = (j + 34 * p) * pixelswidth * 4 + i+ 18 * 4 * k;
                        int indexascii = j * 18 * 4 + i;

                        outputImage[indexpixel] =  Asciimage[asciiImageIndex][indexascii];


                    }
                }
            }
        }

        Log.d(TAG, "End");
        return outputImage;

        // That's the end of YZ's edit  -> End
    }


    //Function: calculate the average value of the region with specified region indexes
    //input: the index of one region
    //output: the average of that region
    public int PixelimageAve(int startcol, int startrow, byte[] pixels) {

        int sum = 0;
        for (int j = startrow; j < startrow + 34; j++) {
            for (int i = startcol; i < startcol + 18 * 4; i=i+4) {


                sum += ( pixels[j * 72 + i + 0] & 0xff); // red
                sum += ( pixels[j * 72 + i + 1] & 0xff); // green
                sum += (( pixels[j * 72 + i +2] & 0xff)); // blue
            }
        }
        int avgpixel =( (sum) / (18 * 34 * 3))+40;

        return avgpixel;
    }
    //Function: calculate the average of an ascii image
    //Input:  the byte array of an ascii image
    //Output: the average value of that image

    public int getAvg(byte[] b) {
        int sum = 0;

        for (int row = 0; row < 34; row++) {
            for (int col = 0; col < 72; col = col + 4) {


                sum += ( b[row * 72 + col + 0] & 0xff); // red
                sum += ( b[row * 72 + col + 1] & 0xff); // green
                sum += (( b[row * 72 + col + 2] & 0xff)); // blue


            }
        }
        int avgascii = (sum) / (18 * 34 * 3);

        return avgascii;
    }
    //Function: find the index of closest average ascii image for each region on the original picture
    //input: index of each region, and original picture byte array
    //output: the index of closest average ascii image
    public int findMin(int startcol1, int startrow1, byte[] pixels) {
        int diff = Math.abs(PixelimageAve(startcol1, startrow1, pixels) - avg_ascii[0]);
        int minindex =0;
        for (int i = 1; i <36; i++) {

            if (diff> Math.abs(PixelimageAve(startcol1, startrow1, pixels) - avg_ascii[i])){
                diff = Math.abs(PixelimageAve(startcol1, startrow1, pixels) - avg_ascii[i]);
                minindex = i;

            }
            else {

            }
        }



        return minindex;

    }

}
```