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

import android.app.Service;
import android.content.Context;
import android.content.Intent;
import android.graphics.Bitmap;
import android.graphics.BitmapFactory;
import android.graphics.Color;
import android.os.Bundle;
import android.os.Environment;
import android.os.Handler;
import android.os.IBinder;
import android.os.MemoryFile;
import android.os.Message;
import android.os.Messenger;
import android.os.ParcelFileDescriptor;
import android.os.RemoteException;
import android.util.Log;

import java.io.ByteArrayInputStream;
import java.io.ByteArrayOutputStream;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.InputStream;
import java.nio.Buffer;
import java.nio.ByteBuffer;
import java.util.ArrayList;
import java.util.Arrays;


public class ArtTransformService extends Service {

    @Override
    public void onCreate(){
        super.onCreate();
        Log.v("test","create");
    }

    public ArtTransformService() {
    }
    String TAG = "ArtTransformService";

    static final int COLOR_FILTER   = 0;
    static final int MOTION_BLUR    = 1;
    static final int SOBEL_EDGE     = 2;
    static final int GAUSSIAN_BLUR  = 3;
    static final int ASCII_ART      = 4;


    private Messenger messenger_2;
    public int img_width;
    public int img_height;
    public int[] args1;
    public float[] args2;


        static {
        System.loadLibrary("my-native-lib");
    }

//    public native String StringFromJNI();
    public native byte[] ColorFilterFromJNI(byte[] b, int[] args);
    public native byte[] GaussianBlurFromJNI(byte[] b, int w, int h, int[] a1, float[] f1);


    AsciiArt mAscii = new AsciiArt(this);
    SobelEdge sobelEdge = new SobelEdge();
    MotionBlur mMB = new MotionBlur();


    class ArtTransformHandler extends Handler{
        @Override

        // Function: handleMessage sent from ArtLib
        // Input: Message.
        //  Output: receive data from library
        public void handleMessage(Message msg){

            Log.d(TAG, "handleMessage(msg)"+ msg.what);
            Bundle dataBundle = msg.getData();
            ParcelFileDescriptor pfd = (ParcelFileDescriptor) dataBundle.get("pfd");
            FileInputStream fios = new FileInputStream(pfd.getFileDescriptor());
            int ind = dataBundle.getInt("index");
            img_width = dataBundle.getInt("width");
            img_height = dataBundle.getInt("height");
            args1 = dataBundle.getIntArray("args1");
            args2 = dataBundle.getFloatArray("args2");

            Log.d(TAG, "The index is " + String.valueOf(ind));
            Log.d(TAG, "The width is " + String.valueOf(img_width));
            Log.d(TAG, "The height is " + String.valueOf(img_height));
            Log.d(TAG, "The intArg is " + String.valueOf(args1[0]));


            byte[] bytes = readFully(fios);
            Log.d(TAG,"colorfilter");
            byte[] processed_bytes = null;

            messenger_2 = msg.replyTo;
            switch (msg.what) {

                case COLOR_FILTER:
                    processed_bytes = ColorFilterFromJNI(bytes,args1);
                    break;
                case MOTION_BLUR:
                    processed_bytes = mMB.motionBlur(bytes,img_width,img_height,args1);

                    break;
                case SOBEL_EDGE:
                    processed_bytes = bmpToByte(sobelEdge.sEdge(byteToBmp(bytes),args1
));
                    break;
                case GAUSSIAN_BLUR:
                    processed_bytes = GaussianBlurFromJNI(bytes,img_width,img_height,
args1,args2);

                    break;
                case ASCII_ART:
                    processed_bytes = mAscii.ascii(bytes);
                    break;

                default:
                    break;
            }

            try {
                // Send back the processed byte array

                Log.d(TAG,"The byte array is " + String.valueOf(processed_bytes));
                MemoryFile memFile_ret = null;
                memFile_ret = new MemoryFile("processed", processed_bytes.length);
                memFile_ret.allowPurging(true); //
                memFile_ret.writeBytes(processed_bytes, 0, 0, processed_bytes.length);

                ParcelFileDescriptor pfd_ret = MemoryFileUtil.getParcelFileDescriptor(
memFile_ret);
                Bundle processedBundle = new Bundle();
                processedBundle.putParcelable("pfd_ret", pfd_ret);

                try {

                    msg.setData(processedBundle);
                    msg.what = 10;
                    messenger_2.send(msg);
                    if(msg == null)
```

```
                        Log.d("msg is null", "null");



                } catch (RemoteException e) {
                    e.printStackTrace();
                }
            } catch (IOException e) {
                e.printStackTrace();
            }

        }

    }


    public byte[] readFully(FileInputStream input)
    {
        byte[] byteArray = null;
        try
        {
            ByteArrayOutputStream bos = new ByteArrayOutputStream();
            byte[] b = new byte[1024*8];
            int bytesRead =0;

            while ((bytesRead = input.read(b)) != -1)
            {
                bos.write(b, 0, bytesRead);
            }

            byteArray = bos.toByteArray();
            Log.d(TAG,"The byte array is " + String.valueOf(byteArray[0]));
        }
        catch (IOException e)
        {
            e.printStackTrace();
        }



        return byteArray;
    }

    public Bitmap byteToBmp (byte[] b){

        Buffer buf = null;
        buf = ByteBuffer.wrap(b);
        Bitmap.Config conf = Bitmap.Config.ARGB_8888;
        Bitmap bmp = Bitmap.createBitmap(img_width, img_height, conf);

        bmp.copyPixelsFromBuffer(buf);

        return bmp;
    }

    public byte[] bmpToByte(Bitmap bitmap){

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

        byte[] bytes = buffer.array();

        return bytes;
    }


    final Messenger mMessenger = new Messenger(new ArtTransformHandler());
    @Override
    public IBinder onBind(Intent intent) {
        // TODO: Return the communication channel to the service.
        return mMessenger.getBinder();
    }
}

```