What has been done:

After the Office hour we outlined this method with which the handwritten text could be identified from the image. I tried to implement it using the steps below:

- Make the image 8-bit
- Split the image to top and bottom
- Take the histogram lists normalize it until a certain point (240 in this case)
- Use the line equation on the image to get rid of the background and change the bottom half of the image as well.
- Subtract the new image from the original one to keep only the "changed" parts of the image which will be the handwritten part.

I tried this method a couple of times, but it was not working for general cases when the printed text was mainly above for like one line, and the handwritten text was most of the page.

I searched for Threshold algorithms and found this one called Otsu's method that computes the threshold and applies it to create a binary image. I wrote the code and applied it to the image, which isolated the printed text. I will include the code in the github as well.

After isolating printed text I tried to manipulate the image a bit. I made it binary, sharpened and then dilated. After eroding and dialting the small parts of the image were erased and we could see the main text.

After that I wrote a code which isolated the handwritten text from the main picture. I also added some parts like cropping and inverting it. After I get the image with only the handwritten part I invert the picture one more time to get the white background and black handwritten text and use the Hough algorithm. The core method where the Hough Transform algorithm is implemented.
It initializes an output space (paramSpace) to store Hough space parameters.It iterates through the input image pixels and performs edge detection (pixels with intensity < 150 are considered edges). For each edge pixel, it computes Hough space parameters (angles and distances) and updates the corresponding accumulator cells in the parameter space. It then normalizes the values in the parameter space to display the accumulator as an image. Finally, it creates and displays an ImagePlus object to show the Hough space. After that I use the threshold method in ImageJ to find the "darkest" spots, which correspond to the most present angles in the image and then with those darkest spots I find the coordinates of the darkest spots, which correspond to the coordinates on the picture (a bit of manipulation needed for this). With those coordinates I cropped the text of that area and used the other code I wrote to take out the text available in that cropped part. All the codes and pictures will be available on Github. Everything is in Analysis folder and Java code is in the image folder in the analysis folder.