RAID Project

Submission Date: 9th of December 11:59 PM

1 Introduction

1.1 Submission

Submit a **folder** that is **only** containing your Java source files (*.java) to the course's Homework folder.

Full path: **F:\COURSES\UGRADS\COMP130\Homework\**

Note: COMP 131 students, please submit to the folder under the COMP 130 directory.

Please use the following naming convention for the submitted folders:

YourPSLetter_CourseCode_Surname_Name_HWNumber_Semester

Example folder names:

- **PSA_COMP130_Surname_Name_HW4_F18**
- **PSB_COMP131_Surname_Name_HW4_F18**

Additional notes:

- Using the naming convention properly is important, failing to do so may be **penalized**.
- **Do not** use Turkish characters when naming files or folders.
- Submissions with unidentifiable names will be **disregarded** completely. (ex. "homework1", "project" etc.)
- Please write your name into the Java source file where it is asked for.
- If you are resubmitting to update your solution, simply append **v#** where # denotes the resubmission version. (i.e. **v2**)

1.2 Academic Honesty

Koç University's *Statement on Academic Honesty* holds for all the homeworks given in this course. Failing to comply with the statement will be penalized accordingly. If you are unsure whether your action violates the code of conduct, please consult with your instructor.

1.3 Aim of the Project

The aim of the project is to allow you to practice knowledge on Strings, Arrays and Image Manipulation. In this project, you are asked to develop a Java program which reads an image data and stores it using arrays with which then you will simulate RAID 0, 1, 5 and 10.

1.4 Given Code

In the code given to you, you will see that there are parts already implemented in the project. **Do not** change anything in the code if it is indicated to you with a comment. The code given to you has something called **JavaDoc** comments above all the methods. These comments allow you to view various information about the method when you mouse over the name of the method. Below are the methods given to you in the code with their explanation.

1.4.1 Given Methods

Your program shall implement and call the following methods:

- private GImage createRaidImage(int raid)
- private void failRaid(int raid)
- private void createrRaid0Discs(GImage image)
- private void createrRaid1Discs(GImage image)
- private void createrRaid10Discs(GImage image)

You can create additional methods. However, you shouldn't replace the above methods with new ones.

1.4.2 Given Constants

Constants are given at the bottom of the project. All constants provide **JavaDoc** comments above them. Please read these to understand what constant is used for what. **Do not** use another variable or a static value for something if there is a constant variable defined for that purpose.

1.5 Further Questions

For further questions **about the project** you may contact **Ayça Tüzmen** at [atuzmen@ku.edu.tr]. Note that it may take up to 24 hours before you receive a response so please ask your questions **before** it is too late. No questions will be answered when there is **less than two days** left for the submission.

2 Project Tasks

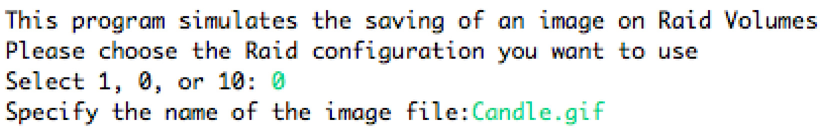
The project consists of 5 parts. In part 1, you should read an image and convert its data into a multidimensional array. In part 2, you are asked to simulate saving the contents of the multidimensional array to imaginary discs using RAID 0, 1 and 10 algorithms. In Part 3, you will simulate the scenario where an imaginary disk crashes and you are expected to access the contents of the remaining discs. In Part 4, using the contents of the remaining discs you will reconstruct the image back again.

2.1 Part 1 - Creating 2D Pixel Data from an Image

In Part 1, you are asked to read an image file, and convert the pixel values of the image into a two-dimensional array.
The program shall ask the user to enter the name of the image file.
It shall prompt user to reenter until a valid file name is entered.
Homework package contains three images that you can work with.
Hint: Use getPixelArray() method.

2.2 Part 2 - Storing in Raid Configuration

The program shall display a menu in the console and allow the user to select the Raid configuration that will be used to store the image data. The menu shall display:



```
This program simulates the saving of an image on Raid Volumes
Please choose the Raid configuration you want to use
Select 1, 0, or 10: 0
Specify the name of the image file:Candle.gif
```

Figure 1: Menu

2.2.1 RAID 0 - Strip Data in Two Discs

You will simulate saving of image data using RAID 0.

RAID 0 allows the data in the RAID volume to be striped across the array's members. Striping divides data into units and distributes those units across the members without creating data redundancy, but improving read/write performance (Figure 2).

Your program shall simulate the saving of the image data using **RAID 0** configuration. The pixel values captured in Part 1, needs to be converted from 2D array (int [][]) into **two separate one dimensional (int[]) arrays**.

Each one dimensional array will simulate a single disc. Your program should **strip pixel values** from the 2D array into two one dimensional arrays.

For example, assume the pixel values of the image data is stored as such in the 2D array as such:
int [][] pixel = {
{-1,-4,-9213102,-10857935,-10857936,-10857937,-10857938},
{-5,-1,-92453102,-12357939,-10567931,-10832932,-10857933} }

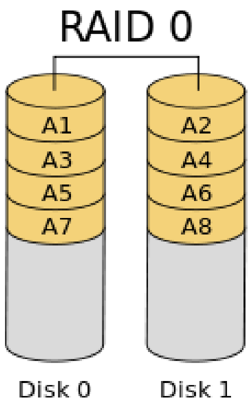


Figure 2: RAID 0.

Your program shall store these pixel values as such:
int [] disc1 =
{-1,-92453102,-10857936,-10857938,-1,-12357939,-10832932 }
int [] disc2 =
{-4,-10857935,-10857937,-5,-92453102,,-10567931,-10857933},

2.2.2 RAID 1 - Mirroring Data in Two Discs

You will simulate saving of image data using *RAID 1*.

RAID 1 allows the data in the RAID volume to be **mirrored** across the RAID array's members. Mirroring is the term used to describe the key feature of RAID 1, which writes duplicate data to each member; therefore, creating data redundancy and increasing fault tolerance (Figure 3).

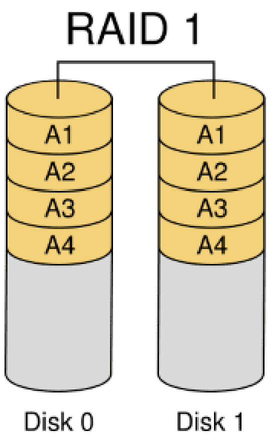


Figure 3: RAID 1.

The program shall simulate saving of the image data using **RAID 1** configuration. The pixel values captured in Part 1, needs to be converted from 2D array (int [][]) into **two separate one dimensional (int[]) arrays**.

Each one dimensional array will simulate a single disc. Your program should **mirror pixel values** from the 2D array into two one dimensional arrays.

For example, assume the pixel values of the image data is stored as such in the 2D array as such:
int [][] pixel = {
{-1,-4,-9213102,-10857935,-10857936,-10857937,-10857938},
{-5,-1,-92453102,-12357939,-10567931,-10832932,-10857933} }

The program shall store these pixel values as such:
int [] disc1 =
{1,-4,-9213102,-10857935,-10857936,-10857937,-10857938,-5,-1,-92453102,-12357939,-10567931,-10832932,-10857933}
int [] disc2 =
{1,-4,-9213102,-10857935,-10857936,-10857937,-10857938,-5,-1,-92453102,-12357939,-10567931,-10832932,-10857933}

2.2.3 RAID 10

The program shall simulate saving of image data using *RAID 10*.

Using *RAID 10*, information is striped across a two disk arrays for system performance. Each of the drives in the arrays has a mirror for fault tolerance. RAID 10 provides the performance benefits of RAID 0 and the redundancy of RAID 1. However, it requires double the hard drives (Figure 4).

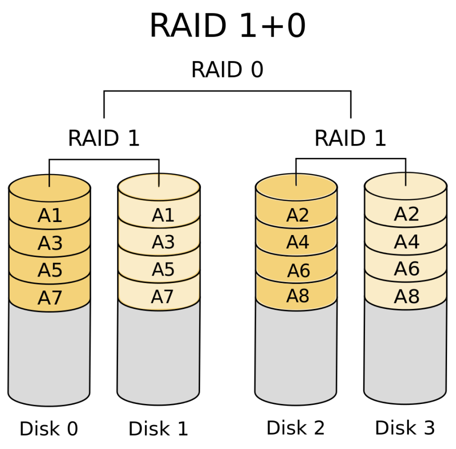


Figure 4: RAID 10.

2.2.4 Task 3 of Task Group 2

Third task. Or underlined.

2.3 Part 3 - Failing a Disc

Your program should simulate the failure of a disc in a Raid configuration. It should **randomly decide** which disc to fail. Once decided on the disc, it will randomly **swap** the content of the failing disc internally.

If the user selected **Raid 0 or Raid 1** as the configuration, your program should fail **one out of the two discs**. (For example, disc 1). It will randomly swap the contents of failing disc internally.

If the user selected Raid 10 as the configuration, it will randomly fail only **one the discs out of four discs** in Raid 10 (e.g. disc 4) . It will randomly swap the contents of failing disc internally.

2.4 Part 4 - Creating a New Image from Raid Discs

Your program shall create a new image from the pixel values in the discs used in a Raid Configuration.

- If the user selected Raid 0, the new image should be created using the pixel values in **two discs (both failing and non-failing discs)**.
- If the user selected Raid 1, the new image should be created using of **one of the discs (non-failing disc)** in Raid 1.
- If the user selected Raid 10, the new image should be created using of **two discs (non-failing discs)** in Raid 10.

2.5 Part 5 - Displaying the original and recovered image

Your program should display the original and the recovered image from Raid configuration on the canvas.
A sample output is shown here.

2.6 End of Project

Your project ends here. You may continue to tinker with the code to implement any desired features and discuss them with your section leader. However, **do not** include any additional features that you implement after this point in to your submission.

Final Warning: Do not include anything beyond this point to your submission. Points may be deducted from your grade.

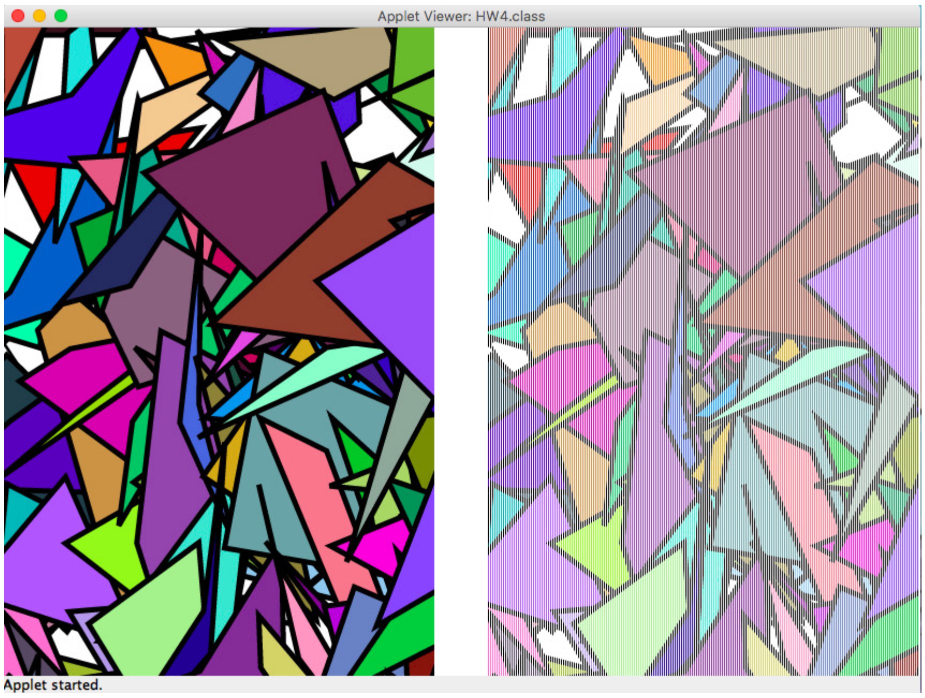


Figure 5: RAID 0 Output.

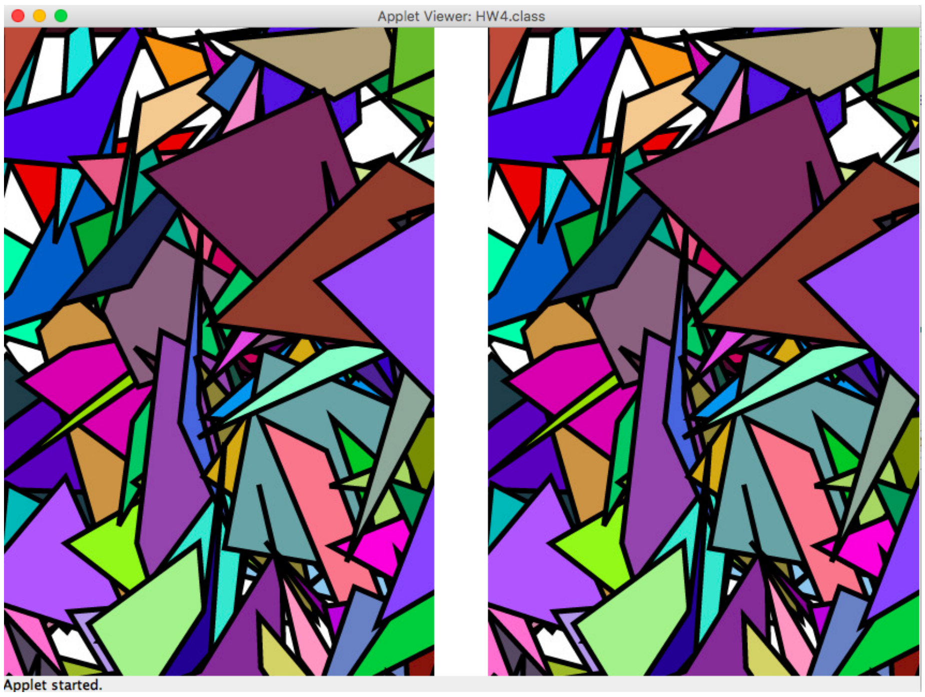


Figure 6: RAID 1 Output.