IMAGE ORGANISER SOFTWARE

MODULE: CSI_5_OOP_2223

DATE OF SUBMISSION: 25/11/2022

STUDENT ID: 4008609

Contents

1. INTRODUCTION

The following report outlines the object-oriented implementation of an image organiser software which sorts files based on the date retrieved (from either Exif, a date within the file name, or file timestamp date The software code outputs a new file (containing subfolders) which lists the sorted files into a particular set of subfolders, the options available to the user, for this output file are; Year, Year and Month, Year, Month and date.

1.1. Aims and objectives

This project aims to convert a procedural-based java program into an object-oriented program. The main objectives being

1) Modularity - easily portable into other software (reusability)

2) Encapsulation – protects variables

3) Clearer readability – allows for easier modifications

4) Easier maintainability - for future debugging

5) Creating a more secure software code that uses encapsulated classes with private variables (and lists) where possible

1.2. Reasoning for private accessors and getters

Though I could have initialised the instance variables with public accessors and avoided the use of getters, it is a better decision to make the fields private and use get methods ensure encapsulation within the class which prevents other classes within the same package (or separate package) from calling or modifying the instance variables. As well, as allowing me to control whether another class has permissions to (along with reading) write to data(using setters), in this use case we do not want the class to be modified at any point within the program, so have not used setters. The implementation of setters and getters allows for easier debugging and verification in the future since we can utilise breakpoints or print statements to see which threads are accessing the fields. Furthermore, you can also put conditions within getters in setters, a simpler case may be if a programmer only wants to allow the user to set specific values to an instance variable i.e., 1 to 100, these conditions and be implemented within the set method.

1.3. Constructors

To allow for objects (instances of a class) to be created within the main class, constructors have been implemented for all classes, though not included in the main body of the report (only in appendix (see section 9)). As per common practice, the constructors are named the same as the class, starting with an upper-case character, while being non-static and not returning any values. It is worthwhile pointing out that if a constructor is not provided by the programmer, the compiler could automatically create a no-argument constructor that does not have any parameters, however, this could obviously cause errors.

1.4 Oracle

1.4.1 APIs

Oracle(2022) states that APIs define the core Java platform for general-purpose computing. These APIs are in modules whose names start with java.

Oracle(2022) explains that JDK APIs are specific to the JDK and may not be available in all implementations of the Java SE Platform.

1.4.2 Tutorials

Oracle(2022) provides groups of lessons that are organized into "trails". These include packages, JAR files and classes.

2. Implementation

2.1 Top level Design decisions and implementations

From the outset, when viewing the original code, there were some things that needed to be changed. There were other items that would benefit from being changed.

A diagram showing the top level changes and implementation is shown in figure 2.1.

One of the items that definitely needed to be changed was the five lists called ‘TargetFolders’. These 5 lists (for folders) contained a list for the folder name, the recursion flag and three other flags to specify how the date is to be “mined”. In an object-oriented environment, it would be much more advantageous to group these 5 lists as a single list of objects. The main advantage is that the five lists will be contained in a single “modular” list allowing for easier troubleshooting/debugging and can be reused in other software. Furthermore, the shift to a single list of objects is much easier to follow and provides greater functionality. Hence, the above is replaced by a class called “FoldersandFlags” (see figure 2.1, below). The “addfolder” method, which was previously a static method (in the original code), has been added as an instance method into this class (and removed as a static method). Furthermore, as the “getfiles” static method (in the original code) only required the ‘targetfolders’ and ‘recurse’ parameters (which are both now contained in this new class with the single list of objects). It makes sense to include the ‘getfile’ static method within the class as an instance method.

The second item which stands out in the original code is the array of objects which is returned in the static methods for both ‘computeDateDestination’ and ‘ParseDateFromFilename’. An array of objects (as used in the original code), is an array of undefined variables. It makes absolutely no sense, either logically or technically, to use an array of objects in an object-oriented environment. Therefore to reflect this, the array of objects has been replaced by a class called ‘DateandMethod’ (see figure 2.1, below) which has three integer variables for the date, one string variable to specify how the date of the file was determined and an additional boolean field has been added to confirm (or deny) that the file date has been found. This class is used for both ‘computeDateDestination’ and ‘ParseDateFromFilename’ thus removing the array of objects for both of these static functions. Again, the ‘computeDateDestination’ and the ‘ParseDateFromFilename’ have been removed as a static method (as in the original code) and have been incorporated into this class as an instance method. The ‘ParseDateFromFilename’ static function is called from the ‘computeDateDestination’, which is now also in the same class.

Some further improvements have been made to other static methods and the main class. These include coding improvements to the static method ‘scan folders’. Resulting in a more efficient and functional loop method for accessing each of the folders (in the list of folders). Also, some coding amendments have been made to the main class to reflect the above.

The design decisions for the above modifications/improvements are discussed in detail in the following sections (see sections/subsections (2.2 to 2.2.3.2) below.

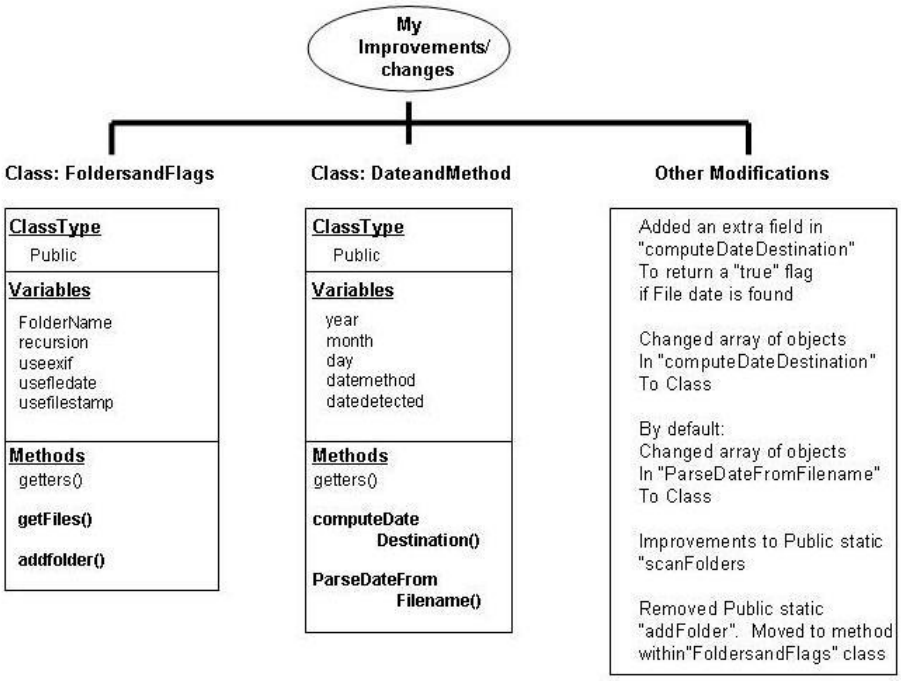

Figure 2.1: Top level design changes and implementation.

2.2 Classes implemented and modified

2.2.1 UML Class diagram

The UML class diagram below shows the structure of the main class and the two additional classes implemented (FoldersandFlags and DatandMethod). The diagrams (See figure 2.2) provide an abstraction of the software as well as giving a detailed insight into the structure of the system. As can be seen, the 'addFolder' and 'getfiles' methods have been included in the 'FolderandFlags' class (as instance methods). Similarly the 'computeDateDestination' and ' 'ParseDateFromFilename' are included in the 'DateandMethod' class (again as instance methods).

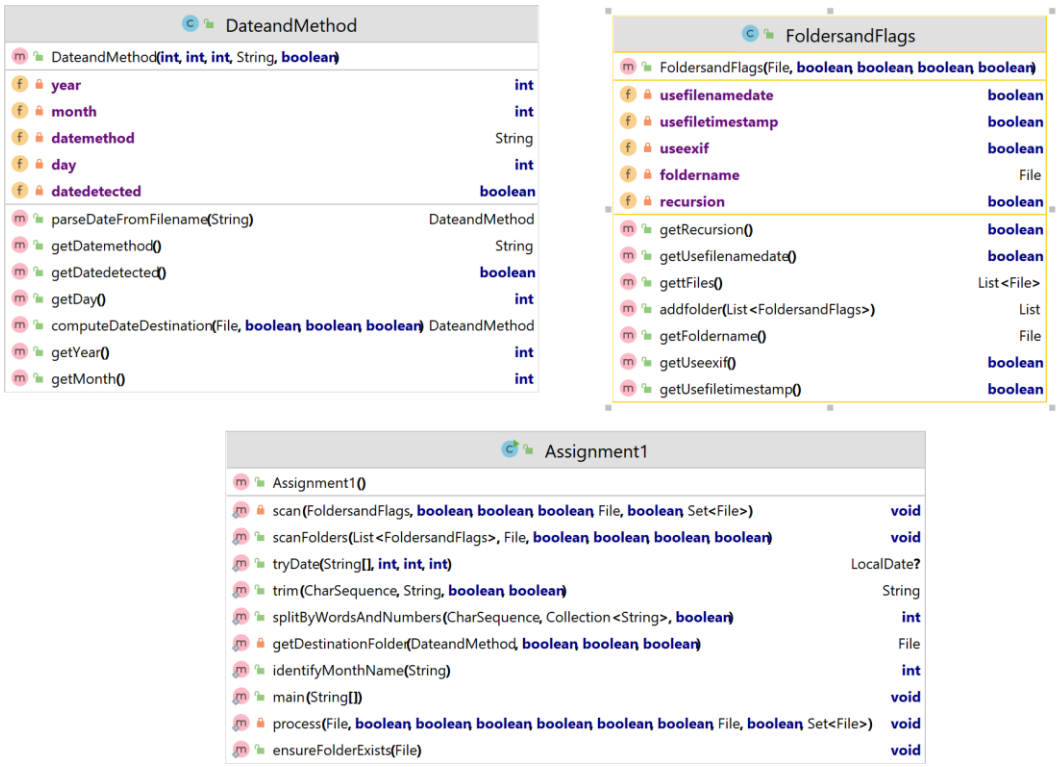

Figure 2.2: UML class diagram (CLASSES: Assignment1, DateandMethod and FoldersandFlags)

2.2.2 CLASS 'FolderssandFlags' – Design and implementation

A simplified code listing for the Class FolderssandFlags is given in figure 2.3 (see below). The full listing is given in the appendix (see section 8)

As stated above (see sec 2.1), the first (and most obvious) candidate for improvement using OOP-based design approach was the five lists called "targetFolders….". Initially, a new class was created containing 5 separate instance variables to replace the first element in the 5 lists. Since the "addfolder" static method (in the original code) uses these same five variables it was decided to include the "addfolder" static method into the class as an instance method (see figure 2.1 – 2.3). As the "addfolder" is an instance method within the class which is also used by other classes, it can be declared as "public" and not "public static". The "addfolder" instance method is used to create (and append) a list of objects. Each object contains the Foldername, a recursion flag and 3 further Boolean flags (see below). The 'addfolder' method returns a list of objects (appended) called 'listoffolders'. 'listoffolders' is declared inside the 'main' method as '*List<FoldersandFlags> listoffolders = new ArrayList<FoldersandFlags>();*'.

Furthermore, the “getfiles” static method in the original code only uses the folder(name) and recurse parameters and both of these are contained in the new single list of objects within the class. It makes sense to also include this method within the class. The “getfiles” static method has been incorporated into the “Foldersandflags” class as an instance method. Because all the parameters required by “getfiles” are already in the class, no parameters need to be passed to the “getfiles” instance method. As before (with the addfolder method), the “getfiles” method is now within the class but is still used by other classes, it can therefore be declared as “public”.

2.2.2.1 Class variables (FoldersandFlags)

The class contains five instance variables which essentially perform the same task as the first element in the five lists (see figure 2.1 – 2.3). These variables are foldername (File) and four booleans’ which are recursion, useexifdate, usefilenamedate and usefiletimestamp. The ‘boolean’ type has been initialised with a lowercase ‘b’ as in this specific use case it would be ideal to utilise it as a primitive rather that an object wrapper. This prevents the possibility of the boolean variables from having null values, as well as decreasing the space taken up in memory on the user’s system. Each time a new folder has added the list of objects (called “listoffolders”) a new instance of these five variables is created.

Five “getters” have also been added to the class so that these five variables may be accessed by another class. Since these variables are accessed by another class, all five “getters” are declared as “public”. “Setters” for this class (variables) are not required anywhere in the code and therefore have not been declared within the class.

2.2.2.2 Methods within the ‘FoldersandFlags’ class

As stated above (see sec 2.3.1), two methods have been incorporated into the class as “instance methods”. These methods are “addfolder” and “getfiles”. The “getfiles” method has been essentially copied from the original code. The ‘addfolders’ method has been created. It returns an appended list of folders. The ‘this’ in the return statement ‘listoffolders.add(this);’ returns the current object instance.

Both methods use variables (and lists) within the class and by doing this the class is “encapsulated perfectly” providing modularity, protection within the class and easier readability Only variables (or lists) that are used by other classes are declared as “public”

```java
package src.oop.cw1_2223.assignment;

import java.io.File;

import java.util.ArrayList;
import java.util.Arrays;
import java.util.List;


public class FoldersandFlags {

    private File foldername;
    private boolean recursion;
    private boolean useexif;
    private boolean usefilenamedate;
    private boolean usefiletimestamp;


    public FoldersandFlags(File foldername, boolean recursion, boolean useexif,
boolean usefilenamedate, boolean usefiletimestamp) {
        this.foldername = foldername;
        this.recursion = recursion;
        this.useexif = useexif;
        this.usefilenamedate = usefilenamedate;
        this.usefiletimestamp = usefiletimestamp;
    }

    public File getFoldername() {
        return foldername;
    }

    public boolean getRecursion() {
        return recursion;
    }
         .    .    .    .

         .    .    .    .  //SEE APPENDIX FOR FULL LISTING

         .    .    .    .




    public  List<File> getFiles() {

         .    .    .    .

         .    .    .    .  //SEE APPENDIX FOR FULL LISTING

         .    .    .    .


         return list;
    }



    public List addfolder(List<FoldersandFlags> listoffolders){

        listoffolders.add(this);

        return listoffolders;
    }

}
```

Figure 2.3: Code snippet (simplified) for class FoldersandFlags

2.2.3 CLASS 'DateandMethod' – Design and implementation

A simplified code listing for the Class DateandMethod is given in figure 2.4 (see below). The full listing is given in the appendix (see section 8)

As stated above (see sec 2.1), the second candidate for improvement using OOP-based design approach was the the array of objects which is returned in the static methods for both 'computeDateDestination' and ParseDateFromFilename.

A new class was created containing 5 instance variables. Four variables replace the four elements in the array (of objects) and an additional field "datedetected" is added as a Boolean flag to indicate that a date (from file) has been detected.

The reason for this extra field "datedetected" is that in the original code, the array was initalised as 'null' prior to searching for a date (using either Exif, Filename or FileTimestamp). Then to check if a date had actually been found, the array was checked to see if it was still 'null'. Here, once an instance of a new class is created it can not easily be set to 'null'. Therefore, this additional field has been added as a boolean flag. Without the use of this additional field the program returned a 'null pointer' error if the object was returned as 'null' in another class and then "polled" using the if statement. The above error could have been overcome by using "exception" commands but it was decided that using an additional field was a good solution.

The last line of the Static method 'ParseDateFromFilename' has been changed from 'return null' to '**return new** DateandMethod(0, 0, 0, **" "**,**false**);' to reflect the above. Also, this new "datedetcected' flag is set to true whenever a date has been detected elsewhere.

Furthermore, the 'computeDateDestination' and 'ParseDateFromFilename' static methods in the original code both originally used the object of array which has now been turned in to a class. Therefore, It makes sense to also include these method within the class. Again the 'computeDateDestination' and 'ParseDateFromFilename' static methods has been incorporated into the "DateandMethod" class as an instance method. Both the 'computeDateDestination' and 'ParseDateFromFilename method are still used by other classes, hence they are declared as "public".

2.2.3.1 Class variables (DateandMethod)

The class contains five instance variables. The first four (year, month, day and datemethod) essentially perform the same task as array of objects (see figure 2.1 – 2.3). The fifth instance variable "datedetected" is a Boolean flag to indicate that a date has actually been detected (see above [sec 2.2.2.1] for detailed explanation). The 'boolean' type has been initialised with a lowercase 'b' as in this specific use case it would be ideal to utilise it as a primitive rather that an object wrapper. This prevents the possibility of the boolean

variables from having null values, as well as decreasing the space taken up in memory on the user's system.

Five "getters" have also been added to the class so that these five variables may be accessed by another class. Since these variables are accessed by another class, all five "getters" are declared as "public". "Setters" for this class (variables) are not required anywhere in the code and therefore have not been declared within the class.

2.2.3.2 Methods within the 'DateandMethods' class

As stated above (see sec 2.3.1), two methods have been incorporated into the class as "instance methods". These methods are 'computeDateDestination' and 'ParseDateFromFilename'.Both methods have been copied from the original code.

The code in both methods has been slightly amended to allow the use of the extra boolean field "datedetected", which could have been replace with "exception" commands but it was decided to use an extra flag instead.

By adding both methods in to the class, the class is "encapsulated" providing modularity, protection and easier readability Only variables (or lists) that are used by other classes are declared as "public"

```
public class DateandMethod {

    private int year;
    private int month;
    private int day;
    private String datemethod;
    private boolean datedetected;

    public DateandMethod(int year, int month, int day, String datemethod,
boolean datedetected) {
        this.year = year;
        this.month = month;
        this.day = day;
        this.datemethod = datemethod;
        this.datedetected=datedetected;
    }

    public int getYear() {
        return year;
    }

    public int getMonth() {
        return month;
    }
        .    .    .

        .    .    . //SEE APPENDIX FOR FULL LISTING


    public DateandMethod computeDateDestination(File file, boolean
useExifDate, boolean useFilenameDate, boolean useFileTimestamp) {

        DateandMethod dateDestination;
        dateDestination = new DateandMethod(0, 0, 0, " ",false);

                .    .    .

                .    .    . //SEE APPENDIX FOR FULL LISTING


            dateDestination = new DateandMethod (zdt.getYear(),
zdt.getMonthValue(), zdt.getDayOfMonth(), "File timestamp",true);
        }
        return dateDestination;
    }



    public DateandMethod parseDateFromFilename(final String filename) {
         .    .    .

         .    .    . //SEE APPENDIX FOR FULL LISTING

          .    .    .
            return new DateandMethod(0, 0, 0, " ",false);
        }
    }


}
```

Figure 2.4: Code snippet (simplified) for class FoldersandFlags

2.2.4 Other design decision and implemenations

Three static methods (namely ‘getfiles’, ‘ComputeDateDestination’ and ‘parseDateFromFilename’) have been removed as static methods and inserted into two separate classes as instance methods.

The ‘addfolder’, which was previously a single line instruction, has been added as an instance method in to the files and flags class.

A new list of objects, to replace Targetfolders, has been declared within the main method as ‘List<FoldersandFlags> listoffolders = new ArrayList<FoldersandFlags>();’. The code listing is shown in figure 2.5 below.

```
376  public static void main(String[] args) throws IOException, ImageProcessingException {
377
378      List<FoldersandFlags> listoffolders = new ArrayList<FoldersandFlags>();
379
380      File commonParent = new File( pathname: "C:\\Users\\irfan\\Desktop\\cwimages"); // change this to point at wherever your cwimages folder is
381
382      File outputFolder = new File(commonParent,  child: "PhotoOrganiserOutput");
383
384      FoldersandFlags n1 = new FoldersandFlags(new File( pathname: commonParent+"\\dsacw_images"),  recursion: false,  useexif: true,  usefilenamedate: true,  usefiletimestamp: true);
385      FoldersandFlags n2 = new FoldersandFlags(new File( pathname: commonParent+"\\oopimages"),  recursion: false,  useexif: true,  usefilenamedate: true,  usefiletimestamp: true);
386
387      n1.addfolder(listoffolders);
388      n2.addfolder(listoffolders);
389
390      boolean simulation = true;
391      boolean useYearFolders = true;
392      boolean useYearAndMonthFolders = true;
393      boolean useYearMonthDayFolders = true;
394
395      scanFolders(listoffolders,outputFolder, simulation, useYearFolders, useYearAndMonthFolders, useYearMonthDayFolders);
396
397  }
398
399
400 }
```

Figure 2.5: Code snippet for main method

Also, other changes to accommodate the new field (‘datedetected’) in the ‘DateandMethod’ class have been made in the new ‘parseDateFromFilename’ instance method.

Coding improvements have also been made to the ‘scan’ method to access variables in getfiles() using the getters inside theFoldersandFlags class. See sigure 2.6 below

```
private static void scan(
        FoldersandFlags foldertoscan,
        final boolean useYearFolders, final boolean useYearAndMonthFolders, final boolean useYearMonthDateFolders,
        File outputFolder, boolean simulation, Set<File> duplicateDetector) throws IOException {

  List<File> files = foldertoscan.getFiles();
  for(File file : files) {
    process(
            file,
            foldertoscan.getUseexif(), foldertoscan.getUsefilenamedate(), foldertoscan.getUsefiletimestamp(),
            useYearFolders, useYearAndMonthFolders, useYearMonthDateFolders,
            outputFolder, simulation, duplicateDetector);
  }
}
```

Figure 2.6: Code snippet for scan method

Also, code improvements have been made to the ‘scanfolders’ method resulting in a more efficient and functional loop method for accessing each of the folders in the ‘list of folders’. The new code listing is shown in figure 2.7 below.

```
33 @  public static void scanFolders(List <FoldersandFlags> listoffolders,
34                                   File outputFolder, final boolean simulation,
35                                   final boolean useYearFolders, final boolean useYearAndMonthFolders, final boolean useYearMonthDateFolders) throws IOException
36      Set<File> duplicateDetector = new HashSet<>();
37
38      for(FoldersandFlags Foldername :listoffolders) {
39        scan(Foldername, useYearFolders, useYearAndMonthFolders, useYearMonthDateFolders, outputFolder, simulation, duplicateDetector);
40      }
41    }
```

Figure 2.7: Code snippet for scanfolders method

3. Testing method and results

3.1 Method(s) of testing

The new code was tested in three different ways.

1) The output folder (“PhotoOrganiserOutput”) generated by the newly modified code was compared to the same folder generated by the original code. The output folder (and subfolders) are compared for both the recursive and non-recursive options (see figure 3.1, 3.2).

2) Secondly, a comparison was made between the print output (generated on the screen) from the newly modified code and that printed from the original code. To keep this document simple, only the printout for the newly modified code is shown. But it was found that the two printouts were identical.

3) Thirdly, a table has been created which shows the file count for each of (and multiple) options (Recurse, Exif/Timestamp/FileName). The files were counted using the console printout. The file count was measured for five different combinations of the recursive, exif, filename and file time stamp flags. The results were compared to the same counts taken after running the original code. A table of results is shown in figure 3.5.

3.2. Results

3.2.1 Tree diagram: Testing with one level (no sub folders i.e recursion = false)

The tree diagram below (see figure 2.8) generated using the command prompt provides a top-down view of the file directories. Two trees are generated, one using the improved new OOP code, and the other using the original procedural based code. Both trees are generated with recursion flag set to false (all other variables true). The results are shown in figure 3.1, below.

As can be seen from both tree diagrams below, both trees are identical.

Improved code

```
Folder PATH listing
Volume serial number is 1C6A-EBE4
C:\USERS\[redacted]\DESKTOP\CWIMAGES
+---dsacw_images
|   +---other
|   |       01-pagodas.jpg
|   |       07-Duke_Humfrey's_Library_Interior_3,_Bodleian_Library,_
                                                    Oxford,_UK_-_Diliff.jpg
|   |
|   \---subfolder
|           06-tianzishan_wulingyuan_zhangjiajie_10Jan2012.jpg
|           HuangShan.jpg
|           Mehlschwalbe_Delichon_urbicum.jpg
|
+---oopimages
|   \---subfolder
|       |   01-pagodas.jpg
|       |   EPP_Congress_Rotterdam_-_Day_1_(52112638468).jpg
|       |   Foxglove2.jpg
|       |
|       \---deeper
|               Gobierno_de_Azerbaiy n,_Baku,_Azerbaiy n,_
                                                    2016-09-26,_DD_27.jpg
|               Holy_Rosary,_Lander,_Wyoming.jpg
|               Julia_2022-10-09_0710Z.jpg
|
\---PhotoOrganiserOutput
    +---2006
    |   +---2006-01
    |   |   \---2006-01-19
    |   |           Plain_tiger_moat.jpg
    |   \---2006-10
    |       \---2006-10-13
    |               KendrewMyoglobin.jpg
    +---2010
    |   \---2010-09
    |       \---2010-09-13
    |               Min_Kyawzwa_Nat.jpg
    +---2012
    |   \---2012-01
    |       \---2012-01-10
    |               02-foxboro20120110.png
    +---2017
    |   \---2017-11
    |       \---2017-11-19
    |               Cardamom_buns.jpg
    \---2022
        \---2022-01
            +---2022-01-24
            |       04-bug.jpg
            |       05-pyramids.jpg
            \---2022-01-31
                    03-Ataleoftwokitties_restored.jpg
                    08-Randell_Cottage_03.jpg
```

Original code

```
Folder PATH listing
Volume serial number is 0000003F 1C6A:EBE4
C:\USERS\[redacted]\DESKTOP\CWIMAGES
+---dsacw_images
|   +---other
|   |       01-pagodas.jpg
|   |       07-Duke_Humfrey's_Library_Interior_3,_Bodleian_Library,
                                                    _Oxford,_UK_-_Diliff.jpg
|   |
|   \---subfolder
|           06-tianzishan_wulingyuan_zhangjiajie_10Jan2012.jpg
|           HuangShan.jpg
|           Mehlschwalbe_Delichon_urbicum.jpg
|
+---oopimages
|   \---subfolder
|       |   01-pagodas.jpg
|       |   EPP_Congress_Rotterdam_-_Day_1_(52112638468).jpg
|       |   Foxglove2.jpg
|       |
|       \---deeper
|               Gobierno_de_Azerbaiy n,_Baku,_Azerbaiy n,_
                                                    2016-09-26,_DD_27.jpg
|               Holy_Rosary,_Lander,_Wyoming.jpg
|               Julia_2022-10-09_0710Z.jpg
|
\---PhotoOrganiserOutput
    +---2006
    |   +---2006-01
    |   |   \---2006-01-19
    |   |           Plain_tiger_moat.jpg
    |   \---2006-10
    |       \---2006-10-13
    |               KendrewMyoglobin.jpg
    +---2010
    |   \---2010-09
    |       \---2010-09-13
    |               Min_Kyawzwa_Nat.jpg
    +---2012
    |   \---2012-01
    |       \---2012-01-10
    |               02-foxboro20120110.png
    +---2017
    |   \---2017-11
    |       \---2017-11-19
    |               Cardamom_buns.jpg
    \---2022
        \---2022-01
            +---2022-01-24
            |       04-bug.jpg
            |       05-pyramids.jpg
            \---2022-01-31
                    03-Ataleoftwokitties_restored.jpg
                    08-Randell_Cottage_03.jpg
```

Figure 3.1: Tree diagram of output directory without recursion

3.2.2 Tree diagram: Testing with multi level (with sub folders i.e recursion = true)

A further test was carried out, this time with recursion set to true (all other variables are also set to true, as before). Two trees were again generated (as before). This time more subfolders have been generated (as expected, due to the recursion set to true). As can be seen from both tree diagrams below (figure 3.2), both trees are again identical.

Improved code

```
Folder PATH listing
Volume serial number is 1C6A-EBE4
C:\USERS\[redacted]\DESKTOP\CWIMAGES
+---dsacw_images
|   +---other
|   \---subfolder
+---copimages
|   \---subfolder
|       |   01-pagodas.jpg
|       |
|       \---deeper
\---PhotoOrganiserOutput
    +---2006
    |   +---2006-01
    |   |   \---2006-01-19
    |   |           Plain_tiger_moat.jpg
    |   \---2006-10
    |       \---2006-10-13
    |               KendrewMyoglobin.jpg
    +---2010
    |   \---2010-09
    |       \---2010-09-13
    |               Min_Kyawzwa_Nat.jpg
    +---2012
    |   \---2012-01
    |       \---2012-01-10
    |               02-foxboro20120110.png
    |               06-tianzishan_wulingyuan_zhangjiajie_10Jan2012.jpg
    +---2013
    |   \---2013-01
    |       \---2013-01-28
    |               Holy_Rosary,_Lander,_Wyoming.jpg
    +---2016
    |   \---2016-09
    |       \---2016-09-26
    |               Gobierno_de_Azerbaiy n,_Baku,_Azerbaiy n,_
                                                2016-09-26,_DD_27.jpg
    +---2017
    |   \---2017-11
    |       \---2017-11-19
    |               Cardamom_buns.jpg
    +---2019
    |   \---2019-03
    |       \---2019-03-10
    |               Foxglove2.jpg
    \---2022
        +---2022-01
        |   +---2022-01-24
        |   |       01-pagodas.jpg
        |   |       04-bug.jpg
        |   |       05-pyramids.jpg
        |   \---2022-01-31
        |           03-Ataleoftwokitties_restored.jpg
        |           07-Duke_Humfrey's_Library_Interior_3,_Bodleian_Library,
                                                    _Oxford,_UK_-_Diliff.jpg
        |           08-Randell_Cottage_03.jpg
        |           HuangShan.jpg
        |           MehlSchwalbe_Delichon_urbicum.jpg
        +---2022-05
        |   \---2022-05-31
        |           EPP_Congress_Rotterdam_-_Day_1_(52112638468).jpg
        \---2022-10
            \---2022-10-09
                    Julia_2022-10-09_0710Z.jpg
```

Original code

```
Folder PATH listing
Volume serial number is 1C6A-EBE4
C:\USERS\[redacted]\DESKTOP\CWIMAGES
+---dsacw_images
|   +---other
|   \---subfolder
+---copimages
|   \---subfolder
|       |   01-pagodas.jpg
|       |
|       \---deeper
\---PhotoOrganiserOutput
    +---2006
    |   +---2006-01
    |   |   \---2006-01-19
    |   |           Plain_tiger_moat.jpg
    |   \---2006-10
    |       \---2006-10-13
    |               KendrewMyoglobin.jpg
    +---2010
    |   \---2010-09
    |       \---2010-09-13
    |               Min_Kyawzwa_Nat.jpg
    +---2012
    |   \---2012-01
    |       \---2012-01-10
    |               02-foxboro20120110.png
    |               06-tianzishan_wulingyuan_zhangjiajie_10Jan2012.jpg
    +---2013
    |   \---2013-01
    |       \---2013-01-28
    |               Holy_Rosary,_Lander,_Wyoming.jpg
    +---2016
    |   \---2016-09
    |       \---2016-09-26
    |               Gobierno_de_Azerbaiy n,_Baku,_Azerbaiy n,_
                                                2016-09-26,_DD_27.jpg
    +---2017
    |   \---2017-11
    |       \---2017-11-19
    |               Cardamom_buns.jpg
    +---2019
    |   \---2019-03
    |       \---2019-03-10
    |               Foxglove2.jpg
    \---2022
        +---2022-01
        |   +---2022-01-24
        |   |       01-pagodas.jpg
        |   |       04-bug.jpg
        |   |       05-pyramids.jpg
        |   \---2022-01-31
        |           03-Ataleoftwokitties_restored.jpg
        |           07-Duke_Humfrey's_Library_Interior_3,_Bodleian_Library,
                                                    _Oxford,_UK_-_Diliff.jpg
        |           08-Randell_Cottage_03.jpg
        |           HuangShan.jpg
        |           MehlSchwalbe_Delichon_urbicum.jpg
        +---2022-05
        |   \---2022-05-31
        |           EPP_Congress_Rotterdam_-_Day_1_(52112638468).jpg
        \---2022-10
            \---2022-10-09
                    Julia_2022-10-09_0710Z.jpg
```

Figure 3.2: Tree diagram of output directory with recursion

3.2.3 Testing by comparing console printouts

Two snippets of the console printout for the new OOP code were generated and are shown in figure 3.3 and 3.4 (see below). The two printouts are generated with the recursive flag set to true and false. All ofther flags (useExifDate, useFilenameDate ,useFileTimestamp, simulation, useYearFolders, useYearAndMonthFolders, useYearMonthDayFolders) set to true in both cases. Although, only printouts for the newly generated are given below, both of these printouts were compared to the printouts generated by the original code and found to be identical.

Recursion = true

Date determined for 'C:\Users\irfan\Desktop\cwimages\dsacw_images\02-foxboro20120110.png' 2012-1-10 (File name)

Renamed 'C:\Users\irfan\Desktop\cwimages\dsacw_images\02-foxboro20120110.png' to 'C:\Users\irfan\Desktop\cwimages\PhotoOrganiserOutput\2012\2012-01\2012-01-10\02-foxboro20120110.png'

Date determined for 'C:\Users\irfan\Desktop\cwimages\dsacw_images\03-Ataleoftwokitties_restored.jpg' 2022-1-31 (File timestamp)

Renamed 'C:\Users\irfan\Desktop\cwimages\dsacw_images\03-Ataleoftwokitties_restored.jpg' to 'C:\Users\irfan\Desktop\cwimages\PhotoOrganiserOutput\2022\2022-01\2022-01-31\03-Ataleoftwokitties_restored.jpg'

Date determined for 'C:\Users\irfan\Desktop\cwimages\dsacw_images\04-bug.jpg' 2022-1-24 (File timestamp)

Renamed 'C:\Users\irfan\Desktop\cwimages\dsacw_images\04-bug.jpg' to 'C:\Users\irfan\Desktop\cwimages\PhotoOrganiserOutput\2022\2022-01\2022-01-24\04-bug.jpg'

Date determined for 'C:\Users\irfan\Desktop\cwimages\dsacw_images\05-pyramids.jpg' 2022-1-24 (File timestamp)

Renamed 'C:\Users\irfan\Desktop\cwimages\dsacw_images\05-pyramids.jpg' to 'C:\Users\irfan\Desktop\cwimages\PhotoOrganiserOutput\2022\2022-01\2022-01-24\05-pyramids.jpg'

Date determined for 'C:\Users\irfan\Desktop\cwimages\dsacw_images\08-Randell_Cottage_03.jpg' 2022-1-31 (File timestamp)

Renamed 'C:\Users\irfan\Desktop\cwimages\dsacw_images\08-Randell_Cottage_03.jpg' to 'C:\Users\irfan\Desktop\cwimages\PhotoOrganiserOutput\2022\2022-01\2022-01-31\08-Randell_Cottage_03.jpg'

Date determined for 'C:\Users\irfan\Desktop\cwimages\dsacw_images\other\01-pagodas.jpg' 2022-1-24 (File timestamp)

Renamed 'C:\Users\irfan\Desktop\cwimages\dsacw_images\other\01-pagodas.jpg' to 'C:\Users\irfan\Desktop\cwimages\PhotoOrganiserOutput\2022\2022-01\2022-01-24\01-pagodas.jpg'

Date determined for 'C:\Users\irfan\Desktop\cwimages\dsacw_images\other\07-Duke_Humfrey's_Library_Interior_3,_Bodleian_Library,_Oxford,_UK_-_Diliff.jpg' 2022-1-31 (File timestamp)

Renamed 'C:\Users\irfan\Desktop\cwimages\dsacw_images\other\07-Duke_Humfrey's_Library_Interior_3,_Bodleian_Library,_Oxford,_UK_-_Diliff.jpg' to 'C:\Users\irfan\Desktop\cwimages\PhotoOrganiserOutput\2022\2022-01\2022-01-31\07-Duke_Humfrey's_Library_Interior_3,_Bodleian_Library,_Oxford,_UK_-_Diliff.jpg'

Date determined for 'C:\Users\irfan\Desktop\cwimages\dsacw_images\subfolder\06-tianzishan_wulingyuan_zhangjiajie_10Jan2012.jpg' 2012-1-10 (File name)

Renamed 'C:\Users\irfan\Desktop\cwimages\dsacw_images\subfolder\06-tianzishan_wulingyuan_zhangjiajie_10Jan2012.jpg' to 'C:\Users\irfan\Desktop\cwimages\PhotoOrganiserOutput\2012\2012-01\2012-01-10\06-tianzishan_wulingyuan_zhangjiajie_10Jan2012.jpg'

Date determined for 'C:\Users\irfan\Desktop\cwimages\dsacw_images\subfolder\HuangShan.jpg' 2022-1-31 (File timestamp)

Renamed 'C:\Users\irfan\Desktop\cwimages\dsacw_images\subfolder\HuangShan.jpg' to 'C:\Users\irfan\Desktop\cwimages\PhotoOrganiserOutput\2022\2022-01\2022-01-31\HuangShan.jpg'

Date determined for 'C:\Users\irfan\Desktop\cwimages\dsacw_images\subfolder\MehlschwalbeDelichon_urbicum.jpg' 2022-1-31 (File timestamp)

Renamed 'C:\Users\irfan\Desktop\cwimages\dsacw_images\subfolder\MehlschwalbeDelichon_urbicum.jpg' to 'C:\Users\irfan\Desktop\cwimages\PhotoOrganiserOutput\2022\2022-01\2022-01-31\MehlschwalbeDelichon_urbicum.jpg'

Date determined for 'C:\Users\irfan\Desktop\cwimages\oopimages\Cardamom_buns.jpg' 2017-11-19 (EXIF data)

Renamed 'C:\Users\irfan\Desktop\cwimages\oopimages\Cardamom_buns.jpg' to 'C:\Users\irfan\Desktop\cwimages\PhotoOrganiserOutput\2017\2017-11\2017-11-19\Cardamom_buns.jpg'

Date determined for 'C:\Users\irfan\Desktop\cwimages\oopimages\KendrewMyoglobin.jpg' 2006-10-13 (EXIF data)

Renamed 'C:\Users\irfan\Desktop\cwimages\oopimages\KendrewMyoglobin.jpg' to 'C:\Users\irfan\Desktop\cwimages\PhotoOrganiserOutput\2006\2006-10\2006-10-13\KendrewMyoglobin.jpg'

Date determined for 'C:\Users\irfan\Desktop\cwimages\oopimages\Min_Kyawzwa_Nat.jpg' 2010-9-13 (EXIF data)

Renamed 'C:\Users\irfan\Desktop\cwimages\oopimages\Min_Kyawzwa_Nat.jpg' to 'C:\Users\irfan\Desktop\cwimages\PhotoOrganiserOutput\2010\2010-09\2010-09-13\Min_Kyawzwa_Nat.jpg'

Date determined for 'C:\Users\irfan\Desktop\cwimages\oopimages\Plain_tiger_moat.jpg' 2006-1-19 (EXIF data)

Renamed 'C:\Users\irfan\Desktop\cwimages\oopimages\Plain_tiger_moat.jpg' to 'C:\Users\irfan\Desktop\cwimages\PhotoOrganiserOutput\2006\2006-01\2006-01-19\Plain_tiger_moat.jpg'

Date determined for 'C:\Users\irfan\Desktop\cwimages\oopimages\subfolder\01-pagodas.jpg' 2022-1-24 (File timestamp)

Did not rename 'C:\Users\irfan\Desktop\cwimages\oopimages\subfolder\01-pagodas.jpg' to 'C:\Users\irfan\Desktop\cwimages\PhotoOrganiserOutput\2022\2022-01\2022-01-24\01-pagodas.jpg' as destination file exists.

Date determined for 'C:\Users\irfan\Desktop\cwimages\oopimages\subfolder\EPP_Congress_Rotterdam_-_Day_1_(52112638468).jpg' 2022-5-31 (EXIF data)

Renamed 'C:\Users\irfan\Desktop\cwimages\oopimages\subfolder\EPP_Congress_Rotterdam_-_Day_1_(52112638468).jpg' to 'C:\Users\irfan\Desktop\cwimages\PhotoOrganiserOutput\2022\2022-05\2022-05-31\EPP_Congress_Rotterdam_-_Day_1_(52112638468).jpg'

Date determined for 'C:\Users\irfan\Desktop\cwimages\oopimages\subfolder\Foxglove2.jpg' 2019-3-10 (EXIF data)

Renamed 'C:\Users\irfan\Desktop\cwimages\oopimages\subfolder\Foxglove2.jpg' to 'C:\Users\irfan\Desktop\cwimages\PhotoOrganiserOutput\2019\2019-03\2019-03-10\Foxglove2.jpg'

Date determined for 'C:\Users\irfan\Desktop\cwimages\oopimages\subfolder\deeper\Gobierno_de_Azerbaiyán,_Baku,_Azerbaiyán,_2016-09-26,_DD_27.jpg' 2016-9-26 (EXIF data)

Renamed 'C:\Users\irfan\Desktop\cwimages\oopimages\subfolder\deeper\Gobierno_de_Azerbaiyán,_Baku,_Azerbaiyán,_2016-09-26,_DD_27.jpg' to 'C:\Users\irfan\Desktop\cwimages\PhotoOrganiserOutput\2016\2016-09\2016-09-26\Gobierno_de_Azerbaiyán,_Baku,_Azerbaiyán,_2016-09-26,_DD_27.jpg'

Date determined for 'C:\Users\irfan\Desktop\cwimages\oopimages\subfolder\deeper\Holy_Rosary,_Lander,_Wyoming.jpg' 2013-1-28 (EXIF data)

Renamed 'C:\Users\irfan\Desktop\cwimages\oopimages\subfolder\deeper\Holy_Rosary,_Lander,_Wyoming.jpg' to 'C:\Users\irfan\Desktop\cwimages\PhotoOrganiserOutput\2013\2013-01\2013-01-28\Holy_Rosary,_Lander,_Wyoming.jpg'

Date determined for 'C:\Users\irfan\Desktop\cwimages\oopimages\subfolder\deeper\Julia_2022-10-09_0710Z.jpg' 2022-10-9 (File name)

Renamed 'C:\Users\irfan\Desktop\cwimages\oopimages\subfolder\deeper\Julia_2022-10-09_0710Z.jpg' to 'C:\Users\irfan\Desktop\cwimages\PhotoOrganiserOutput\2022\2022-10\2022-10-09\Julia_2022-10-09_0710Z.jpg'

Process finished with exit code 0

Figure 3.3: Console output for improved code (recursion = true)

Recursion = false

```
Date determined for 'C:\Users\irfan\Desktop\cwimages\dsacw_images\02-foxboro20120110.png' 2012-1-10 (File name)

Renamed 'C:\Users\irfan\Desktop\cwimages\dsacw_images\02-foxboro20120110.png' to
'C:\Users\irfan\Desktop\cwimages\PhotoOrganiserOutput\2012\2012-01\2012-01-10\02-foxboro20120110.png'

Date determined for 'C:\Users\irfan\Desktop\cwimages\dsacw_images\03-Ataleoftwokitties_restored.jpg' 2022-1-31 (File timestamp)

Renamed 'C:\Users\irfan\Desktop\cwimages\dsacw_images\03-Ataleoftwokitties_restored.jpg' to
'C:\Users\irfan\Desktop\cwimages\PhotoOrganiserOutput\2022\2022-01\2022-01-31\03-Ataleoftwokitties_restored.jpg'

Date determined for 'C:\Users\irfan\Desktop\cwimages\dsacw_images\04-bug.jpg' 2022-1-24 (File timestamp)

Renamed 'C:\Users\irfan\Desktop\cwimages\dsacw_images\04-bug.jpg' to
'C:\Users\irfan\Desktop\cwimages\PhotoOrganiserOutput\2022\2022-01\2022-01-24\04-bug.jpg'

Date determined for 'C:\Users\irfan\Desktop\cwimages\dsacw_images\05-pyramids.jpg' 2022-1-24 (File timestamp)

Renamed 'C:\Users\irfan\Desktop\cwimages\dsacw_images\05-pyramids.jpg' to
'C:\Users\irfan\Desktop\cwimages\PhotoOrganiserOutput\2022\2022-01\2022-01-24\05-pyramids.jpg'

Date determined for 'C:\Users\irfan\Desktop\cwimages\dsacw_images\08-Randell_Cottage_03.jpg' 2022-1-31 (File timestamp)

Renamed 'C:\Users\irfan\Desktop\cwimages\dsacw_images\08-Randell_Cottage_03.jpg' to
'C:\Users\irfan\Desktop\cwimages\PhotoOrganiserOutput\2022\2022-01\2022-01-31\08-Randell_Cottage_03.jpg'

Date determined for 'C:\Users\irfan\Desktop\cwimages\oopimages\Cardamom_buns.jpg' 2017-11-19 (EXIF data)

Renamed 'C:\Users\irfan\Desktop\cwimages\oopimages\Cardamom_buns.jpg' to
'C:\Users\irfan\Desktop\cwimages\PhotoOrganiserOutput\2017\2017-11\2017-11-19\Cardamom_buns.jpg'

Date determined for 'C:\Users\irfan\Desktop\cwimages\oopimages\KendrewMyoglobin.jpg' 2006-10-13 (EXIF data)

Renamed 'C:\Users\irfan\Desktop\cwimages\oopimages\KendrewMyoglobin.jpg' to
'C:\Users\irfan\Desktop\cwimages\PhotoOrganiserOutput\2006\2006-10\2006-10-13\KendrewMyoglobin.jpg'

Date determined for 'C:\Users\irfan\Desktop\cwimages\oopimages\Min_Kyawzwa_Nat.jpg' 2010-9-13 (EXIF data)

Renamed 'C:\Users\irfan\Desktop\cwimages\oopimages\Min_Kyawzwa_Nat.jpg' to
'C:\Users\irfan\Desktop\cwimages\PhotoOrganiserOutput\2010\2010-09\2010-09-13\Min_Kyawzwa_Nat.jpg'

Date determined for 'C:\Users\irfan\Desktop\cwimages\oopimages\Plain_tiger_moat.jpg' 2006-1-19 (EXIF data)

Renamed 'C:\Users\irfan\Desktop\cwimages\oopimages\Plain_tiger_moat.jpg' to
'C:\Users\irfan\Desktop\cwimages\PhotoOrganiserOutput\2006\2006-01\2006-01-19\Plain_tiger_moat.jpg'


Process finished with exit code 0
```

Figure 3.4: Console output for improved code (recursion = false)

3.2.4 Testing using the file count method

To ensure the accuracy of this file count method, results were taken by copying a test was carried out by pasting the print output (from console) into a word document and carrying out the 'find' function to determine the number occurrence of keywords (i.e File Name) within the output. The results were then entered into a table for easier readability and compared to similar results generated using the original code.

The tabulated results are shown below (See figure 3.5). As can been seen, the results from the new improved OOP code are identical to those of the original code.

	recursion= true Use Exifdate= true Use Filenamedate= true Usu Filetimestamp= true	recursion= false Use Exifdate= true Use Filenamedate= true Usu Filetimestamp= true	recursion= true Use Exifdate= true Use Filenamedate= false Usu Filetimestamp= false	recursion= true Use Exifdate= false Use Filenamedate= true Usu Filetimestamp= false	recursion= true Use Exifdate= false Use Filenamedate= false Usu Filetimestamp= true
Original Code					
No of files using "useExifdate"	8	4	8	0	0
No of files using "Use Filenamedate"	3	1	0	4	0
No of files using "Use Filetimestamp"	9	4	0	0	20
Improved OOP code					
No of files using "useExifdate"	8	4	8	0	0
No of files using "Use Filenamedate"	3	1	0	4	0
No of files using "Use Filetimestamp"	9	4	0	0	20

Figure 3.5: Table showing file count for both modified and original code

4. Conclusion

4.1 Implementation

Two new classes (FoldersandFlags and DateandMethod) have been implemented successfully.

Two instance methods (namely getfiles() and addfolders()) have been included in the FoldersandFlags class as instance methods. Two further methods (ComputeDateDestination and ParseDatefromFilename) have also been included into the 'DateandMethod' class successfully. Hence the object of array could be permanently deleted from both of these methods. An object array is very similar to an untyped variable which results in the code needing to know what the datatype of each element within the array is (nasty in an OOP environment).

Other changes include an additional field in the 'DateandMethod' class called 'datedetected' which is a boolean flag to indicate when a date has been found. This solves the problem of a null pointer error (described in detail in section 2.2.3). Exception commands could have been used but it was decided to include this additional field.

Additional code improvements were also made successfully to the 'scan' and 'scanfolder' methods. Resulting in a more efficient and functional loop method for accessing each of the folders in the 'list of folders'.

All of the above has been done successfully and results in a much better modular Oject oriented design which is easily maintainable, protected by encapsulated classes, easily readable and can be upgraded in the future

4.2 Testing and results

Only some of the variants have been shown in all three testing methods (i.e comparing outpur folder, comparing the output console printout and counting the number of files in Exif, Filename, Filetimestamp).

This (only some variants) was done for the purpose to simplify this document. several other variants were also tested and found to be identical.

From this, we can assume/comclude that both codes (original and modified) give the same output folder, same printout and the hence the same file count for all variations.

This confirms that the code was converted successfully.

5. Reflecting on learning

5.1 Future improvements to code

The 'FoldersandFlags' class could be split in to an outer class and an inner class. The inner class would house the three boolean flags (useExif, Filenamedate and Filetimestamp).

Also a hierarchical abstract class could be used for the getDateDestination method which goes through three different flags (useExif, useFilenamedate and useFiletimestamp) in "hierarchical order".

5.3 Other improvements

The code could be improved to accept more than two initial folders (currently only dsacw_images and oopimages). This could be done by either asking the user for the number of folders or even having a list of folders at the start. Also in the current code the source directory for the source folder is fixed, it would be helpful for the user so specify the source directory.

Another improvement would be to add the months of the year in different languages (possible using a dictionary). Currently this is only done in enthish in the Filenamedate method.

5.4 Development of own knowledge

Prior to constructing this report, I had experience with programming in OOP, which allowed mini-projects to run without error, however, I did not understand the implications (benefits) of a class-based design, upon converting a procedural program to an OOP design and conducting further research, it is clear how creating classes as reusable components that can be implemented in various programs in the future benefits all programmers in the future who may want to use said classes, or modify/debug them. I now understand the difference between static methods and instance methods, and also the difference between static variables and instance variables

I appreciate the advantages of OOP programming environment including modularity, maintenance, readability and also portability (ability to use in other software).

6. Table of Figures

7. References

[1] Oracle https://docs.oracle.com/en/java/javase/11/docs/api/ (2022) [Accessed 18 November 2022]

[2] Oracle https://docs.oracle.com/javase/8/docs/api/ (2022) [Accessed 18 November 2022]

[3] Oracle https://docs.oracle.com/javase/tutorial (2022) [Accessed 18 November 2022]

8. Bibliography

[1] Oracle https://docs.oracle.com/en/java/javase/11/docs/api/ (2022) [Accessed 18 November 2022]

[2] Oracle https://docs.oracle.com/javase/8/docs/api/ (2022) [Accessed 18 November 2022]

[3] Oracle https://docs.oracle.com/javase/tutorial (2022) [Accessed 18 November 2022]

[4] Mike, CM. (2022) 01- Classes-and-Objects-1 [Object Oriented Programming, CSI_5_OOP_2223]. London south bank university, week 1.

[5] Mike, CM. (2022) 01- Classes-and-Objects-2 [Object Oriented Programming, CSI_5_OOP_2223]. London south bank university, week 2

[6] Mike, CM. (2022) Extension, inheritance, and polymorphism [Object Oriented Programming, CSI_5_OOP_2223]. London south bank university, week 3

[7] Mike, CM. (2022) Abstract Classes, Interfaces and UML [Object Oriented Programming, CSI_5_OOP_2223]. London south bank university, week 4

9. Appendix

9.1 CLASS Assignment 1

```
package src.oop.cw1_2223.assignment;

import com.drew.imaging.ImageProcessingException;

import java.io.File;
import java.io.IOException;
import java.time.DateTimeException;
import java.time.LocalDate;
import java.util.ArrayList;
import java.util.Collection;
import java.util.HashSet;
import java.util.List;
import java.util.Set;

public class Assignment1 {
  //private static List<FoldersandFlags> listoffolders = new ArrayList<FoldersandFlags>();


  /**
   * Scans all the folders that have been added to the target folders list, determining
   * the appropriate destination date and moving the files to their new locations - unless
   * simulation is true, in which it just prints what moves would be made but does not
   * do it.
   *
   * @param useYearFolders
   * @param useYearAndMonthFolders
   * @param useYearMonthDateFolders
   * @throws IOException
   */



  public static void scanFolders(List <FoldersandFlags> listoffolders,
                                 File outputFolder, final boolean simulation,
                                 final boolean useYearFolders, final boolean useYearAndMonthFolders, final boolean useYearMonthDateFolders) throws IOException {
    Set<File> duplicateDetector = new HashSet<>();

    for(FoldersandFlags Foldername :listoffolders) {
      scan(Foldername, useYearFolders, useYearAndMonthFolders, useYearMonthDateFolders, outputFolder, simulation, duplicateDetector);
    }
  }




  /**
   * Scans the files in the target folder identified by the index i in the target folder list.
   *
   * @param foldertoscan
   * @param useYearFolders
   * @param useYearAndMonthFolders
   * @param useYearMonthDateFolders
   * @param simulation
   * @param duplicateDetector
   * @throws IOException
   */
  private static void scan(
          FoldersandFlags foldertoscan,
          final boolean useYearFolders, final boolean useYearAndMonthFolders, final boolean useYearMonthDateFolders,
          File outputFolder, boolean simulation, Set<File> duplicateDetector) throws IOException {

    List<File> files = foldertoscan.getFiles();
    for(File file : files) {
      process(
              file,
              foldertoscan.getUseexif(), foldertoscan.getUsefilenamedate(), foldertoscan.getUsefiletimestamp(),
              useYearFolders, useYearAndMonthFolders, useYearMonthDateFolders,
              outputFolder, simulation, duplicateDetector);
    }
  }


```

```
73
74    /**
75     * Process and move the given file.
76     *
77     * @param file
78     * @param useExifDate
79     * @param useFilenameDate
80     * @param useFileTimestamp
81     * @param useYearFolders
82     * @param useYearAndMonthFolders
83     * @param useYearMonthDateFolders
84     * @param simulation
85     * @param duplicateDetector
86     * @throws IOException
87     */
88    private static void process(
89            File file, boolean useExifDate, boolean useFilenameDate, boolean useFileTimestamp,
90            boolean useYearFolders, boolean useYearAndMonthFolders, boolean useYearMonthDateFolders,
91            File outputFolder, boolean simulation, Set<File> duplicateDetector) throws IOException {
92
93      DateandMethod dateDestination = new DateandMethod(0, 0, 0, " ",false);
94      dateDestination = dateDestination.computeDateDestination(file, useExifDate, useFilenameDate, useFileTimestamp);
95
96
97      if (dateDestination.getDatedetected()==true) {
98        System.out.println("Date determined for '" + file + "' " + dateDestination.getYear() + "-" + dateDestination.getMonth() + "-" + dateDestination.getDay() + " (" + dateDestination.getDatemethod() + ")");
99        File destinationFolder = getDestinationFolder(dateDestination, useYearFolders, useYearAndMonthFolders, useYearMonthDateFolders);
100       File outputDestinationFolder = new File(outputFolder, destinationFolder.getPath());
101       File destinationFilename = new File(outputDestinationFolder, file.getName());
102
103       if (simulation) {
104         if (duplicateDetector.add(destinationFilename)) {
105           System.out.println( "[SIMULATING] Renaming '" + file + "' to '" + destinationFilename + "'");
106         } else {
107           System.out.println( "[SIMULATING] Did not rename '" + file + "' to '" + destinationFilename + "' as destination file exists.");
108         }
109       } else {
110         ensureFolderExists(outputDestinationFolder);
111         if (!destinationFilename.exists()) {
112           if (file.renameTo(destinationFilename)) {
113             // issue confirmation of file move
114             System.out.println( "Renamed '" + file + "' to '" + destinationFilename + "'");
115           } else {
116             // issue warning that file move failed
117             System.out.println( "Failed to rename '" + file + "' to '" + destinationFilename + "'");
118           }
119         } else {
120           // an attempt to move second file to same name in same date folder
121           // issue warning and do not move
122           System.out.println( "Did not rename '" + file + "' to '" + destinationFilename + "' as destination file exists.");
123         }
124       }
125     } else {
126       // issue warning that no date could be determined for file
127       System.out.println( "Could not determine date for '" + file + "'.");
128     }
129   }
130
131
132
133   /**
134    * Converts a YMD date held as Integer objects in the dateDestination array into a folder
135    * path including levels of containing folders as specified by the other arguments.
136    * For example, with all boolean arguments true, we might return the path 2020/2020-05/2020-05-02,
137    * while with only useYearAndMonthFolders true we would just return 2020-05.
138    *
139    * @param dateDestination
140    * @param useYearFolders
141    * @param useYearAndMonthFolders
142    * @param useYearMonthDateFolders
143    * @return
144    */
145   private static File getDestinationFolder(
146           final DateandMethod dateDestination,
147           final boolean useYearFolders,
148           final boolean useYearAndMonthFolders,
149           final boolean useYearMonthDateFolders) {
150     File parent = null;
151     if (useYearFolders) {
152       parent = new File(parent, String.format("%04d", (Integer)dateDestination.getYear()));
153     }
154     if (useYearAndMonthFolders) {
155       parent = new File(parent, String.format("%04d", (Integer)dateDestination.getYear()) + "-" + String.format("%02d", (Integer)dateDestination.getMonth()));
156     }
157     if (useYearMonthDateFolders) {
158       parent = new File(parent, String.format("%04d", (Integer)dateDestination.getYear()) + "-" + String.format("%02d", (Integer)dateDestination.getMonth())  + "-" + String.format("%02d", (Integer)dateDestination.getDay()));
159     }
160     return parent;
161   }
162
163
164
165   /**
166    * Check a folder path exists and create it with all required folders if it does not.
167    * <ul>
168    * <li>If folder is an existing directory, returns.
169    * <li>If folder does not exist, attempts to create it and throws an exception on failure.
170    * <li>If folder is a file, throws an exception.
171    * </ul>
172    * If this method returns, folder exists and is a folder.
173    *
174    * @param folder
175    * @throws IOException
176    */
177   public static void ensureFolderExists( final File folder) throws IOException {
178     if (folder.exists()) {
179       if (folder.isDirectory()) {
180         return;
181       } else {
182         throw new IOException( "Folder is actually a file: " + folder);
183       }
184     } else {
185       ensureFolderExists(folder.getParentFile());
186       if (folder.mkdir()) {
187         return;
188       } else {
189         throw new IOException( "Could not create folder: " + folder);
190       }
191     }
192   }
```

```java


  /**
   * The fragments string array is assumed to contain numerical strings, one representing
   * the year, one a month and one a day. Which is which is determined by the year, month
   * and day parameters (for example, if year = 0, then fragments[0] is used to determine
   * the year value). After discarding any leading zeros from the fragments an attempt is
   * made to construct a valid date from them, and if this succeeds that date will be
   * returned. Otherwise, null will be returned. For example calling this with
   * ["2001", "05", "01"], 0, 1, 2 will return a date representing 01 May 2001.
   * On the other hand ["2001", "25", "01"], 0, 1, 2 will return null as 25 is not a valid
   * month.
   *
   * @param fragments
   * @param year
   * @param month
   * @param day
   * @return
   */
  public static LocalDate tryDate(String[] fragments, int year, int month, int day) {
    for (int j = 0; j < fragments.length; j++) {
      fragments[j] = trim( fragments[j], "0", true, false);
    }
    if (fragments[year].length() == 4 && fragments[month].length() <= 2 && fragments[day].length() <= 2) {
      final int[] n = new int[3];
      for(int j = 0; j < 3; j++) {
        n[j] = Integer.parseInt( fragments[j]);
      }
      try {
        return LocalDate.of(n[year], n[month], n[day]);
      } catch(final DateTimeException x) {
        return null;
      }
    }
    return null;
  }



  /**
   * If fragment is the name of a month or a three letter abbreviation of the
   * name of the month returns a number between 0 (for January) and 11 (for December).
   * If fragment is not recognised returns -1.
   *
   * @param fragment
   * @return
   */
  public static int identifyMonthName(String fragment) {
    fragment = fragment.toLowerCase();
    String[] names = new String[] {
            "January", "February", "March", "April",
            "May", "June", "July", "August",
            "September", "October", "November", "December" };
    for (int i = 0; i < names.length; i++) {
      String name = names[i].toLowerCase();
      if (fragment.length() == 3) {
        name = name.substring(0,3);
      }
      if (name.equals(fragment)) {
        return i;
      }
    }
    return -1;
  }



  /**
   * Generalised trim function allowing the removal of any number of an arbitrary set of chars from
   * either the beginning or end - or both - of a given string.
   *
   * @param target the String to trim
   * @param chars all characters contained in this String will be removed from the affected ends of {@code target}
   * @param leading if true the beginning of target is trimmed
   * @param trailing if true the end of target is trimmed
   * @return the trimmed result
   */
  public static String trim( final CharSequence target, final String chars, final boolean leading, final boolean trailing) {
    final StringBuilder builder = new StringBuilder( target);
    if ( leading) {
      while( (builder.length() > 0) && (chars.indexOf( builder.charAt(0)) != -1)) { builder.delete(0, 1); }
    }
    if ( trailing) {
      while( (builder.length() > 0) && (chars.indexOf( builder.charAt(builder.length() - 1)) != -1)) { builder.delete(builder.length() - 1, builder.length()); }
    }
    return builder.toString();
  }



  /**
   * Splits {@code target} into fragments which are words, numbers or other chararcters and adds them
   * to {@code toAddTo} in the order they occur in {@code target}.
   * <p>
   * Words are defined as contiguous sequences of characters for which {@code Character.isLetter()} returns true,
   * and numbers are defined as contiguous sequences of characters for which {@code Character.isDigit()}
   * returns true. If {@code allowPointInNumbers} is true, the character '.' is treated exactly as if it was a
   * digit. Although this will allow a decimal number to be detected and extracted it will also extract
   * individual or grouped '.' characters (that is ".34....32.122...4." would be extracted as a single token).
   *
   * @param target string to split
   * @param toAddTo collection to add the extracted substrings to
   * @param allowPointInNumbers if true '.' is treated as a digit
   * @return number of words and numbers extracted.
   */
```

```
  public static int splitByWordsAndNumbers( final CharSequence target, final Collection<String> toAddTo, final boolean allowPointInNumbers) {
    final StringBuilder builder = new StringBuilder();
    final int LETTERS = 2;
    final int DIGITS = 1;
    final int OTHER = 0;
    int gatheringMode = OTHER;
    int count = 0;
    int index = 0;
    while( index < target.length()) {
      final char c = target.charAt( index++);
      if ( Character.isLetter( c)) {
        switch ( gatheringMode) {
          case LETTERS:
            builder.append(c);
            break;
          case DIGITS:
          case OTHER:
            if ( builder.length() > 0) {
              toAddTo.add( builder.toString());
              count++;
            }
            builder.setLength( 0);
            builder.append( c);
            gatheringMode = LETTERS;
            break;
          default:
            assert( false);
            break;
        }
      } else if ( Character.isDigit( c) ||( allowPointInNumbers && ( c == '.'))) {
        switch ( gatheringMode) {
          case DIGITS:
            builder.append(c);
            break;
          case LETTERS:
          case OTHER:
            if ( builder.length() > 0) {
              toAddTo.add( builder.toString());
              count++;
            }
            builder.setLength( 0);
            builder.append( c);
            gatheringMode = DIGITS;
            break;
          default:
            assert( false);
            break;
        }
      } else {
        switch ( gatheringMode) {
          case DIGITS:
          case LETTERS:
            if ( builder.length() > 0) {
              toAddTo.add( builder.toString());
              count++;
            }
            builder.setLength( 0);
            builder.append( c);
            gatheringMode = OTHER;
            break;
          case OTHER:
            builder.append(c);
            break;
          default:
            assert( false);
            break;
        }
      }
    }
    if ( builder.length() > 0) {
      toAddTo.add( builder.toString());
      count++;
    }
    return count;
  }


  public static void main(String[] args) throws IOException, ImageProcessingException {

    List<FoldersandFlags> listoffolders = new ArrayList<FoldersandFlags>();

    File commonParent = new File("C:\\Users\\irfan\\Desktop\\cwimages"); // change this to point at wherever your cwimages folder is

    File outputFolder = new File(commonParent, "PhotoOrganiserOutput");

    FoldersandFlags n1 = new FoldersandFlags(new File(commonParent+"\\dsacw_images"), false, true, true, true);
    FoldersandFlags n2 = new FoldersandFlags(new File(commonParent+"\\oopimages"), false, true, true, true);

    n1.addfolder(listoffolders);
    n2.addfolder(listoffolders);

    boolean simulation = true;
    boolean useYearFolders = true;
    boolean useYearAndMonthFolders = true;
    boolean useYearMonthDayFolders = true;

    scanFolders(listoffolders,outputFolder, simulation, useYearFolders, useYearAndMonthFolders, useYearMonthDayFolders);

  }


}
```

9.1 CLASS FoldersandFlags

```java
package src.oop.cw1_2223.assignment;

import java.io.File;

import java.util.ArrayList;
import java.util.Arrays;
import java.util.List;


public class FoldersandFlags {

    private File foldername;
    private boolean recursion;
    private boolean useexif;
    private boolean usefilenamedate;
    private boolean usefiletimestamp;


    public FoldersandFlags(File foldername, boolean recursion, boolean useexif, boolean usefilenamedate, boolean usefiletimestamp) {
        this.foldername = foldername;
        this.recursion = recursion;
        this.useexif = useexif;
        this.usefilenamedate = usefilenamedate;
        this.usefiletimestamp = usefiletimestamp;
    }

    public File getFoldername() {
        return foldername;
    }

    public boolean getRecursion() {
        return recursion;
    }

    public boolean getUseexif() {
        return useexif;
    }

    public boolean getUsefilenamedate() {
        return usefilenamedate;
    }

    public boolean getUsefiletimestamp() {
        return usefiletimestamp;
    }


    /**
     * Get a list of all files and folders in folder, recursing into subfolders if recurse is true.
     *
     * @param foldername
     * @param recursion
     * @return
     */
    public  List<File> getFiles() {
        List<File> list = new ArrayList<File>();
        list.add(foldername);
        int position = 0;
        while (position < list.size()) {
            File current = list.get(position);
            if (current.isDirectory()) {
                list.remove(position); // remove directories from the list to be returned
                if (list.size() == 0 || recursion) {
                    File[] files = current.listFiles();
                    if (files != null) {
                        // files should not be null for a directory, but might be if we do not have read permission
                        list.addAll(Arrays.asList(files)); // add contained files and directories to end of list
                    }
                }
            } else {
                position++; // leave a file in the list and look at the next
            }
        }
        return list;
    }



    public List addfolder(List<FoldersandFlags> listoffolders){

        listoffolders.add(this);

        return listoffolders;
    }

}
```

9.1 CLASS DateandMethod

```java
package src.oop.cw1_2223.assignment;

import com.drew.imaging.ImageProcessingException;
import src.oop.cw1_2223.assignment.ExifUtils;
import static src.oop.cw1_2223.assignment.Assignment1.splitByWordsAndNumbers;
import static src.oop.cw1_2223.assignment.Assignment1.identifyMonthName;
import static src.oop.cw1_2223.assignment.Assignment1.trim;
import static src.oop.cw1_2223.assignment.Assignment1.tryDate;

import java.io.File;
import java.io.IOException;
import java.time.DateTimeException;
import java.time.Instant;
import java.time.LocalDate;
import java.time.ZoneId;
import java.time.ZonedDateTime;
import java.time.format.DateTimeFormatter;
import java.time.format.DateTimeParseException;
import java.util.ArrayList;
import java.util.List;



public class DateandMethod {

    private int year;
    private int month;
    private int day;
    private String datemethod;
    private boolean datedetected;

    public DateandMethod(int year, int month, int day, String datemethod, boolean datedetected) {
        this.year = year;
        this.month = month;
        this.day = day;
        this.datemethod = datemethod;
        this.datedetected=datedetected;
    }

    public int getYear() {
        return year;
    }

    public int getMonth() {
        return month;
    }

    public int getDay() {
        return day;
    }

    public String getDatemethod() {
        return datemethod;
    }

    public boolean getDatedetected() {
        return datedetected;
    }



    /**
     * Attempts to determine the date to use for the given file, using some combination of looking
     * at the EXIF data, parsing the file name and using the file timestamp. The Object array returned
     * will contain the year, month and day values as Integers and a String in the fourth element that
     * tells where the value were found. If no valid date can be determined, null is returned.
     *
     * @param file
     * @param useExifDate
     * @param useFilenameDate
     * @param useFileTimestamp
     * @return
     */
```

```java
    public DateandMethod computeDateDestination(File file, boolean useExifDate, boolean useFilenameDate, boolean useFileTimestamp) {

        DateandMethod dateDestination;
        dateDestination = new DateandMethod(0, 0, 0, " ",false);

        if (useExifDate) {
            try {
                java.util.Date date = ExifUtils.getFirstDate(src.oop.cw1_2223.assignment.ExifUtils.getMetaData(file));
                if (date != null) {
                    ZonedDateTime zdt = date.toInstant().atZone(ZoneId.systemDefault());//.toLocalDate();
                    dateDestination = new DateandMethod (zdt.getYear(), zdt.getMonthValue(), zdt.getDayOfMonth(), "EXIF data",true);
                }
            } catch(IOException x) {
                System.out.println(x);
            } catch (ImageProcessingException x) {
                System.out.println(x);
            }
        }
        if (dateDestination.getDatedetected() == false && useFilenameDate) {
            dateDestination = parseDateFromFilename(file.getName());
        }
        if (dateDestination.getDatedetected() == false && useFileTimestamp) {
            long timestamp = file.lastModified();
            ZonedDateTime zdt = ZonedDateTime.ofInstant(Instant.ofEpochMilli(timestamp), ZoneId.systemDefault());
            dateDestination = new DateandMethod (zdt.getYear(), zdt.getMonthValue(), zdt.getDayOfMonth(), "File timestamp",true);
        }
        return dateDestination;
    }



    /**
     * Try and find a single parseable date in the filename. If none can be found,
     * or more than one can be found this returns null. Otherwise it returns an
     * Object array containing the year, month and day as Integer objects in the
     * first three elements, and the String "File name" in the fourth element, to
     * indicate the date value was found in the file name.
     *
     * @param filename
     * @return
     */
    public DateandMethod parseDateFromFilename(final String filename) {
        final List<LocalDate> possibles = new ArrayList<>();
        final ArrayList<String> fragments = new ArrayList<String>();
        splitByWordsAndNumbers( filename, fragments, false);
        for (int i = 0; i < fragments.size(); i++) {
            final String fragment = fragments.get(i);
            final char ch = fragment.charAt(0);
            boolean keep = true;
            if (Character.isLetter(ch)) {
                final int mi = identifyMonthName(fragment);
                if (mi < 0) {
                    // not a month, discard fragment
                    keep = false;
                } else {
                    // month name identified; replace with number
                    fragments.set(i, String.valueOf(mi + 1));
                }
            } else if (!Character.isDigit(ch)) {
                keep = false;
            } else if (ch == '0') {
                // discard strings consisting only of zeroes
                keep = trim( fragment, "0", true, false).length() != 0;
            }
            if (!keep) {
                fragments.remove(i); i--;
            }
        }
        // we now only have digit fragments
        // first, look for eight digit fragments
        for (int i = 0; i < fragments.size(); i++) {
            final String fragment = fragments.get(i);
            if (fragment.length() == 8) {
                try {
                    LocalDate date = LocalDate.parse(fragment, DateTimeFormatter.BASIC_ISO_DATE);
                    possibles.add(date);
                } catch( final DateTimeParseException x) {}
            }
        }
        // now look for triplets that form valid dates
        String[] temp = new String[3];
        for (int i = 0; i < fragments.size() - 2; i++) {
            fragments.subList(i, i + 3).toArray(temp);
            LocalDate date = tryDate(temp, 0, 1, 2); // year month day
            if (date != null) {
                possibles.add(date);
            }
            date = tryDate(temp, 2, 1, 0); // day month year
            if (date != null) {
                possibles.add(date);
            }
        }
        if (possibles.size() == 1) {
            LocalDate date = possibles.get(0);
            return new DateandMethod (date.getYear(), date.getMonthValue(), date.getDayOfMonth(), "File name",true);


        } else {
            return new DateandMethod(0, 0, 0, " ",false);
        }
    }


}

```