

**Sven Brotkorb
Matrikel 10786473
Projekt Software Engineering
(DLMCSPSE_D)**

Portfolio Teil 1 und 2

- 1. Projektdokumentation**
- 2. Anforderungsdokument**
- 3. Spezifikationsdokument**
- 4. Architekturdokument und Design**

GitHub Repository:

<https://github.com/sbro-test/timestamper-iu.git>

Teil 1 - Projektdokumentation

1. Projektübersicht

Timestamps, also die Zeitstempel mit dem Änderungsdatum von Dateien sind eine wichtige Information, die dennoch von modernen Systemen (Filesystemen/Explorern) eher stiefmütterlich behandelt werden.

Verschiedene Plattformen haben verschiedene Meinungen, wie Uhrzeiten an Sommerzeit-Winterzeit-Grenzen zu behandeln sind, verschiedene Meinungen, welche Zeitzone einem externen Medium zugrunde liegt und MS Windows hat sogar verschiedene Meinungen von dem gleichen Dateimaterial auf FAT32 und NTFS Laufwerken.

Noch schlimmer wird die Situation mit der Einbindung von mobilen Plattformen oder dem Transport von einer Plattform auf die andere.

Die Konsequenz ist, dass man auf modernen Systemen relativ sicher sein kann, dass die Integrität von Dateinamen und Dateiinhalten gewahrt bleibt (was man zudem gut mit Checksummen absichern kann), wohingegen die Zeitstempel der Dateien ein schwer kontrollierbares und undurchschaubares Verhalten zeigen, also salopp gesagt "machen was sie wollen".

Dazu kommt die schlechte Darstellung der Zeitstempel.

Die Explorer der meisten modernen GUI Systeme (MS Windows, Mac, Linux) versuchen sich an lächerlichen Abstraktionen und zeigen z.B. für Dateistempel von heute nur die Uhrzeit, von gestern nur das Wort "Gestern", dann Dateien von diesem Jahr ohne Jahreszahl und dann noch ältere Dateien mit vollem Datum aber ohne Uhrzeit.

Und dann haben neuere MS Windows Versionen im Explorer eine Datums-Spalte, die noch nicht einmal genau sagt, auf welches Datum der Datei sie sich bezieht (erzeugt, geändert, Bild aufgenommen) und die man eher als Zufallsgenerator sehen kann.

Das hier vorgestellte Tool "Timestamper" soll dem entgegenwirken und ein für alle Mal einen sicheren Stand herstellen, indem es:

1. Zeitstempel zunächst einmal sekundengenau anzeigt,
was auf modernen Systemen nur mit den wenigsten Tools zu bewerkstelligen ist
2. die Zeitstempel aus mehreren Quellen gegeneinander abgleichen kann;
wichtigste Kandidaten dafür:
 - zuerst natürlich der Zeitstempel aus dem Filesystem selbst
 - besonders wichtig die verschiedenen Sichten von MS Windows/Linux
 - Handy-Bilder und -Videos, bei denen Datum und Zeit im Dateiname enthalten sind

3. und diese schließlich in einem privaten TEXTFILE speichert/"einfriert", von wo es die Zeitstempel auch restaurieren kann, so dass diese von da an gegen die Kapriolen der Filesysteme geschützt sind.

Eine besondere Motivation für das Tool sind auf dem Computer gespeicherte und von mehreren Handies/Digitalkameras zusammengeführte Bilder, weil hier die Abstimmung der Zeitstempel und deren Zuverlässigkeit besonders wichtig sind.

Übrigens haben manche Bilddateien zwar als Metadaten ein in der Datei gespeichertes Aufnahmedatum, das ist aber aus diversen Gründen leider keine hinreichende Lösung (nur für wenige Dateiformate, eventuell falsche Zeiteinstellung bei der Aufnahme, Änderung bedeutet Änderung des Dateiinhalts).

Das Tool wird bewusst „timezone agnostic“ gestaltet, verarbeitet also reine Timestamps ohne Zeitzone.

Denn erstens verwalten nur die wenigsten Filesysteme außer Timestamps auch Zeitzonen und zweitens geht es bei diesem Programm um eine Desktop-Anwendung für die Verwaltung eigener Datei-/Bildersammlungen und nicht um ein weltweit erreichbares Serversystem.

Es ist also genau unerwünscht, dass die beim Dinner in Sidney um 20:00 Uhr aufgenommen Bilder jetzt zurück in Deutschland den Stamp von 05:00 morgens tragen.

Das Programm wird in Python für aktuelle Betriebssysteme erstellt, zunächst für MS Windows. Zur Erstellung der GUI dient eine moderne GUI Library wie wxWidgets (wxPython) oder Qt for Python.

Zum organisatorischen Ablauf noch ein Hinweis in eigener Sache.

Das Projekt "Software Engineering" (DLMCSPSE_D) hat das erklärte Ziel, das zuvor in anderen Modulen Gelernte in der Praxis anzuwenden.

Bedauerlicherweise sind diese anderen Module überhaupt nicht Teil meiner Weiterbildung bei der IU Akademie. Dies bitte ich zu berücksichtigen.

2.Risikomanagement

1.RISIKO: Kein passendes Control/Widget für die Dateitabelle in der GUI Library

Das zentrale GUI Element des Programms ist eine Tabelle mit den Datei-Metadaten und insbesondere Timestamps aus einem Verzeichnis.

Es ist daher zu hoffen, dass die verwendete GUI Library ein möglichst gut passendes Tabellen-Widget dafür bietet.

Probability/Eintrittswahrscheinlichkeit: GERING

Die derzeit erwogenen GUI Libraries wxWidgets und Qt sind reife und weit entwickelte Softwareprodukte. Ich hatte persönlich vor etwa einem Jahrzehnt Erfahrungen mit einer Art Tabellen-Widget in wxWidgets.

Bei einem Fehlschlag in wxWidgets bleibt auch noch der Ausweg von diesem Open Source Produkt hin zur Qt for Python Community Lizenz, weil Qt im Zweifelsfall "reichere" Widgets bietet.

Severity/Schwere der Auswirkungen: ENORM

Ohne ein passendes Tabellen-Widget müsste die gesamte Tabellen-Funktionalität "von Hand" in einem ganz allgemeinen Canvas-Widget oder Text-Widget nachgebaut werden, was

- 1.schlechter aussieht und funktioniert und
- 2.den Zeitplan ganz erheblich gefährdet.

2.RISIKO: Undurchschaubares Verhalten der Zeitstempel auf Windows und Linux sowie unter verschiedenen Filesystemen

Probability/Eintrittswahrscheinlichkeit: MITTEL

Die Tatsache, dass die diversen Systeme sich so undurchschaubar verhalten, ist ja genau die Motivation für dieses Programm.

Es bleibt aber zu hoffen, dass das Verhalten nichtsdestotrotz Regeln folgt, auch wenn diese seltsam sein mögen.

Severity/Schwere der Auswirkungen: ZEITVERLUST

Irgendwie müssen die Zeitstempel der diversen System ja zu durchschauen sein.

Das Problem besteht im Zeitverlust an Entwicklungszeit für die zahlreichen Experimente und Vergleiche.

3.RISIKO: Technische Schwierigkeiten mit Low Level Funktionen

- die aktuelle Plattform MS Windows oder Linux ist leicht herauszufinden
- aber wie ist die Abfrage des verwendeten Filesystems ?
insbesondere für non-admin User ?
- muss/kann man auf Linux zusätzlich auch die mount-Optionen abfragen ?

Probability/Eintrittswahrscheinlichkeit: GERING

Severity/Schwere der Auswirkungen: GERING

Dies ist nur ein kleiner Teil des Programms, das Programm wäre auch ohne diese Funktionalität absolut nützlich - im Extremfall könnte man sogar ein Dialogfeld vorsehen, wo einige dieser Einstellung von Hand vorgenommen werden.

KEIN ODER MINIMALES RISIKO: (nur zu Vollständigkeit)

- Textzeilen und Buttons in der GUI
- ls/stat Auslesen der Verzeichnisinhalte
- Verwaltung Lesen/Schreiben/Format des eigenen Textfiles
- Datum mit RE aus dem Filename extrahieren

3.Zeitplanung

Planung für einen MONAT Arbeitszeit, in Arbeitswochen/-tagen.

Eine datumsgenaue Planung ist nicht möglich, weil im Portfolio-Projekt mehrere Zwischenschritte vorgesehen sind, an denen auf Feedback gewartet wird; diese Zeit muss mit Lernstoff aus einem anderen Modul überbrückt werden.

1.WOCHE: Phase 1

- 2 TAGE Ideensammlung und "Wunschliste" für das Projekt aus diversen eigenen Quellen und kleinen Experimenten
- 2 TAGE TEXTING für die 3 Schritte der Phase 1
und somit die 3 Teile des einzureichenden PDFs
- 1 TAG Formalitäten, Office Formatierung, Bedienung Pebblepad

2.WOCHE: Phase 2 / PART Implementierung

- 2 TAGE GUI Library und einfache Tabellenanzeige
 - Auswahl und Installation der GUI Library (wxPython oder Qt for Python ?)
 - Überblick über verfügbare Widgets, insbesondere die TABELLEN

- die ersten Inhaltsspalten mit naivem Timestamp

=> ZIEL ist, nach 2 Tagen mit der GUI Library den simpel formatierten Inhalt eines per Parameter genannten Verzeichnisses als Tabelle anzuzeigen.

1 TAG die klassische Timestamp-Spalte aus dem Filesystem,
inklusive Win/Linux Variation

1 TAG die Timestamp-Spalte "aus Filename" als Datenquelle
(die Verwendung dieser Spalte als Ziel einer Operation müsste zu einer Datei-Umbenennung bei Beibehalt des Pattern führen und ist aktuell noch nicht vorgesehen)

1 TAG die Timestamp-Spalte, die für Bindung an das private Sicherungsfile steht
(Speichern ins File, Auslesen aus File, Abgleich)

3.WOCHE: Phase 2 / Überlauf, organisatorischer Teil

2 TAGE die große Übertragsoperation
Quellspalte => Zielspalte mit den Widgets
unten unter der Tabelle

(organisatorischer Teil, Nachtrag in Phase 2)

1 TAG UML Toolchain und UML malen

1 TAG Text schreiben auf Basis UML und gesammelten Notizen

1 TAG Office und Abstract schreiben, letzte Tests

1 TAG Feinschliff, Upload Git+PebblePad

4.WOCHE: Phase 3, insbesondere Testing

(Nachtrag in Phase 2)

3 TAGE Testing und QS

- Exception Handling verbessern
- Hi-Level Testing 2 Sticks alter Stil, neuer Stil,
Meinung von 3 Plattformen Linux/Old Win/New Win
- Meinung von timestampper auf allen 3x3
- ggf. Unit Tests fuer basis.py

1 TAG Texting

Benutzeranleitung, grosses Abstract, Lessons Learned
aus gesammelten Notizen

1 TAG Office

1 TAG Feinschliff, Upload Git+PebblePad

Teil 2 - Anforderungsdokument

1. Stakeholder/Zielgruppe

Zielgruppe sind Profinutzer von MS Windows/Linux/Mac, die sich an der unsauberen Verwaltung und Darstellung der Timestamps von Dateien stören und einen Ausweg suchen - insbesondere Profinutzer, die die verschiedenen Plattformen im Wechsel nutzen oder häufig Daten aus mehreren Quellen zusammenführen.

Die avisierte Hauptanwendung sind Bilder von Digitalkameras und Handies, weil da der Zeitstempel der Aufnahmen sehr wichtig ist, um die Bilder verschiedener Quellen gegeneinander abzulegen und dann gemeinsam in der richtigen Reihenfolge zu betrachten.

Allerdings ist das Programm in keiner Weise auf Bilder und Videos eingeschränkt.

2. Funktionale Anforderungen

1. GENAUE ANZEIGE

Das Programm bietet jeweils eine Tabellenansicht eines Verzeichnisses, die prinzipiell aus der Tabellenansicht des Explorers bekannt ist, allerdings reduziert auf die Kernfunktionen des Programms:

- Name
- Extension
- viele Timestamps (siehe unten)

aber für diese Kernfunktionen erweitert um:

- * prinzipiell sekundengenaue Zeitanzeige
 - * alle Timestamps in ISO-Reihenfolge vom großen zu kleinen Zeitfeld (vom Jahr bis zur Sekunde) in einem zentral konfigurierten Format
 - * außerdem (quasi als Nebeneffekt) bytengenaue Anzeige der Dateigröße
-
- + zudem mehrere Timestamp-Spalten aus verschiedenen Quellen für die Manipulation durch das Programm, siehe unten

1b. Reduzierte Anzeige für UNTERVERZEICHNISSE

Im aktuell gewählten Verzeichnis vorhandene Unterverzeichnisse werden zur Vollständigkeit der Liste und zur Navigation dorthin natürlich mit angezeigt.

Allerdings werden in der Tabelle keine Metadaten für Verzeichnisse angezeigt, insbesondere keinerlei **TIMESTAMPS**.

Die Art und Weise wie Timestamps von Verzeichnissen von den diversen Betriebssystemen gehandhabt werden, ist leider zu unzuverlässig, um sie in irgendeiner Weise sinnvoll zu verwenden.

Nützlich wäre in einer zukünftigen Programmversion beim Verzeichnis allerdings eine Abstraktion auf den neuesten Timestamp aller Dateien darin, das ist nur potentiell sehr zeitaufwendig (z.B. fürs ganze Verzeichnis C:\Windows).

2. VERZEICHNIS-NAVIGATION

Das Programm versucht nicht einen kompletten Explorer-Nachbau, bietet aber natürlich einige Wege, das aktuell angezeigte Verzeichnis zu wechseln:

- Adresszeile oben über der Tabelle
- "Nach oben" [...] Button oben neben der Adresszeile
- Doppelklick auf Unterverzeichnisse in der Anzeige für den Wechsel in das jeweilige Verzeichnis

3. BINDUNG an EXPLORER

Nachdem das Programm nicht die Ambition hat, einen kompletten Explorer nachzubauen, ist die Bindung an den Explorer der jeweiligen Plattform (Thunar, Nemo, Nautilus, Finder ...) für die eventuell wechselseitige Arbeit in beiden Programmen besonders wichtig.

Die Schnittstelle zum jeweiligen Explorer ist dabei prinzipiell der Austausch des Verzeichnis-Pfades im Format der Plattform, z.B.

"D:\Daten\Bilder\212 Urlaub Marokko Ostern 24"
"/mnt/FotoPlatte/212 Urlaub Marokko Ostern 24"

Der Austausch erfolgt in der GUI durch Copy&Paste in die Adresszeile oben über der Tabelle, die natürlich bei Navigation innerhalb des Programms mitgeführt/aktualisiert wird.

4. Verwendung von SORTIERUNG und MARKIERUNG in den Dateilisten

SORTIERUNG: wie im Explorer führt ein Klick auf die verschiedenen Spaltenüberschriften zu einer Sortierung der Liste nach den Werten dieser Spalte

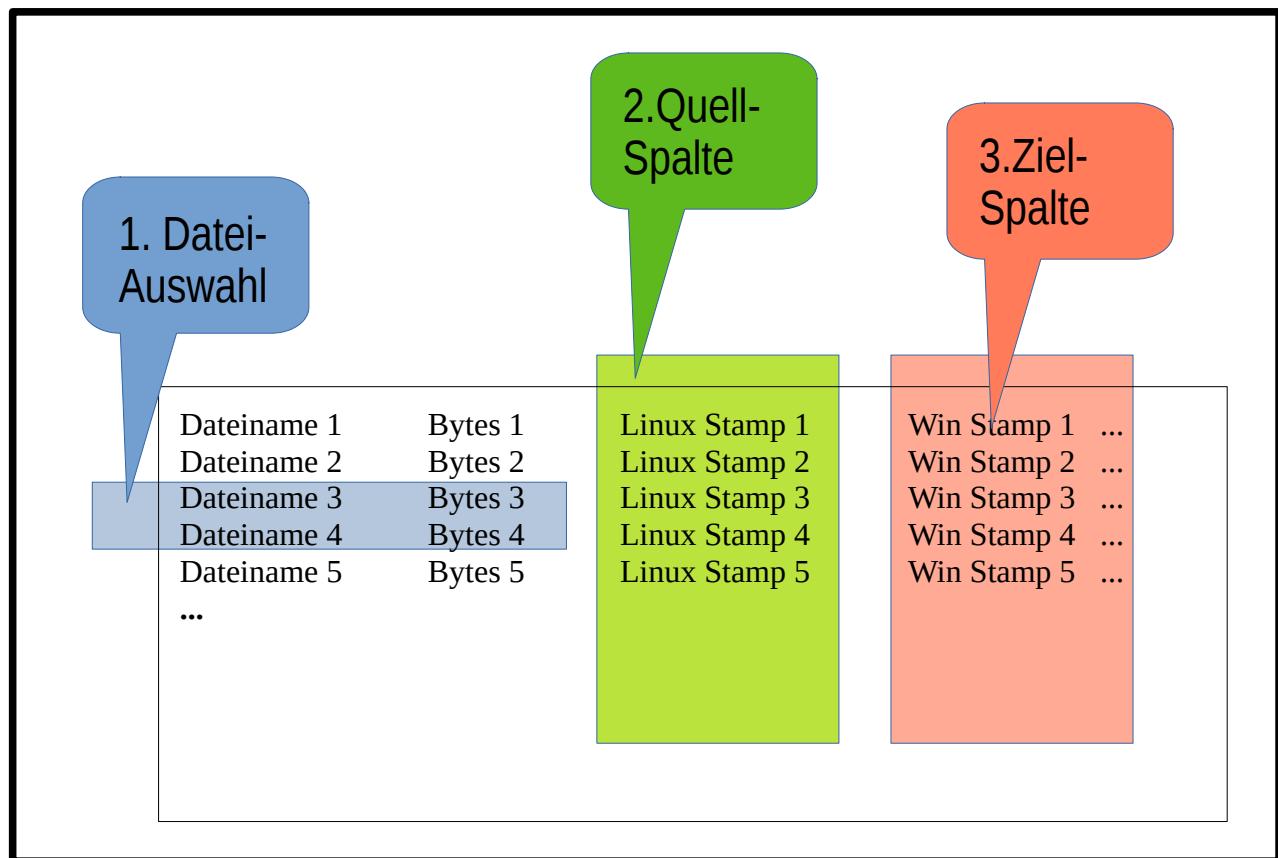
MARKIERUNG: die Markierung der Dateien für die aktuelle anstehende Operation ist ein wichtiges Feature; für die Bedienung der Markierung und die visuelle Darstellung kann man sich hoffentlich auf den Support der GUI Library verlassen

Achtung: Im Gegensatz zu neueren MS Windows Versionen soll die sorgfältig von Hand gemachte Markierung von Dateien unbedingt eine Umsortierung nach anderen Spalten überleben, das ist ein wichtiges Feature für die gezielte Auswahl von Dateien.

5.HAUPTFUNKTION: Abgleich von Timestamps

HAUPTFUNKTION des Programms ist der Abgleich von Timestamp-Werten aus verschiedenen Quellen, was durch den Übertrag von Timestamps aus einer Quellspalte an eine Zielspalte realisiert wird.

Für die Ausführung dieser Operation müssen eine oder mehrere Dateien,
genau eine Quellspalte und
genau eine Zielspalte markiert sein, ...



... dann kann per Button der Übertrag gestartet werden.

Natürlich müssen dafür pro Datei auch mehrere Timestamps aus verschiedenen Quellen verfügbar sein und angezeigt werden...

Im Weiteren folgen jetzt die verschiedenen Timestamp Spalten, die angezeigt werden und die als Quelle und eventuell Ziel für Übertrags-Operationen zur Verfügung stehen:

6.Timestamp-Spalte "modification time" (Python/Linux)

Die wichtigste Timestamp-Spalte ist die "modification time" aus dem Filesystem, und zwar zunächst in Linux/Python Sicht mit guter Sommerzeit Behandlung (siehe Entwurf).

- das ist Timestamp-Spalte 1/4 in der Anzeige,
sie ist als Quelle und Ziel von Übertragungsoperationen verfügbar

7.Timestamp-Spalte "modification time" (MS Windows)

Da bereits bekannt ist, dass die geplante Programmierumgebung Python auch unter MS Windows die Timestamps nach Linux Manier anzeigt, braucht es zumindest für die Nutzung unter Windows (und optional später auch anderswo) auch eine Timestamp Spalte in Windows Explorer Sicht.

- das ist Timestamp-Spalte 2/4 in der Anzeige,
sie ist ebenfalls als Quelle und Ziel von Übertragungsoperationen verfügbar

Für diese beiden Spalten 1/4 und 2/4 ist zu beachten, dass eine Änderung der einen Spalte immer auch die jeweils andere ändert, weil sie ja nur verschiedene Sichten auf den gleichen Speicherort "modification time" im Filesystem sind.

8.Timestamp aus Filename

Eine exotischere Funktion ist die Gewinnung von Timestamps aus den Filenamen. Das ist nützlich, weil Handies inzwischen meist eine Art ISO-Datum als Filename für Bilder und Videos verwenden, was eine besonders gute und vor allem stabile Quelle für Timestamps darstellt.

BSP: 20251224_120000.jpg
Screenshot_20251224_120000_Fennec.png

- das ist Timestamp-Spalte 3/4 in der Anzeige,
sie ist als Quelle von Übertragungsoperationen verfügbar
(eine Verwendung als Ziel von Übertragungsoperationen würde eine Umbenennung von Dateien unter Beibehaltung des Namensmusters bedeuten und ist ein Feature für zukünftige Programmversionen)

9. Timestamp aus privatem Textfile

Neben genauer Anzeige und Übertrag ist die Speicherung der gewünschten (und zuvor genau gesäuberten) Timestamps in einer stabilen Textform eine Kernfunktion des Programms.

Das Programm soll sich aber nicht anmaßen, diese Timestamps ungefragt ins Filesystem zu übertragen oder am Ende einen eventuell unsauberer Stand festzuschreiben.

Vielmehr ist der Timestamp aus den eigenen Textfile eine weitere Timestamp-Spalte für den gezielten, manuellen Übertrag.

- das ist Timestamp-Spalte 4/4 in der Anzeige,
sie ist wieder als Quelle und Ziel von Übertragungsoperationen verfügbar

Eventuell könnte das Programm am Ende (oder bei Verzeichniswechsel ?) fragen, ob ins Textfile zurückgespeichert werden soll, das müssen Experimente und erste Bedienerfahrungen ergeben.

10. Behandlung fehlender Timestamps

Die Datumsspalten 3/4 aus Filename und 4/4 aus Textfile haben die Besonderheit, dass nicht für jedes File ein Timestamp vorliegen wird:

- Stamp 3/4 aus Filename: ein Stamp ist hier sogar eher die Ausnahme als die Regel; er ist eigentlich nur für Bilder und Videos von neueren Handies vorhanden, nicht für Digitalkameras und bestimmt nicht für sonstige Dateien
- Stamp 4/4 aus Textfile: und es wird anfangs auch noch keine von diesem Programm in Textfiles gespeicherten Timestamps geben, derartige Daten besitzt man erst nach längerer Nutzung

Für diese Fälle (kein Timestamp für diese Spalte und diese Datei) wird eine einfache Ersatzanzeige gezeigt, beim Übertrag werden die Werte ignoriert (siehe unten).

+ Phase 2: Erweiterte ANFORDERUNGEN siehe unten...

Erweiterte ANFORDERUNGEN, Zusatz-FEATURES aus dem Entwicklungsprozess siehe letzter Punkt in Phase 2 "Architekturdokument und Design".

3.Nichtfunktionale Anforderungen

1.Sicherheit: keine Änderung von Dateiinhalten

Es werden keine Dateiinhalte geändert (außer dem privaten Textfile des Programms selbst) .

Das Programm kann also verwendet werden, ohne die Änderung der eigentlichen Inhalte zu befürchten; nur Timestamps werden manipuliert.

2.Vollständige Anzeige

Das Programm zeigt alle Dateien und Unterverzeichnisse des aktuell gewählten Verzeichnisses (ob "alle" auch versteckte/hidden Dateien umfasst, wird im Entwurf entschieden) und für diese Dateien alle oben genannten Informationen (insbesondere eben Timestamps).

3.Geschwindigkeit

Es ist normal, dass die Anzeige eines großen Verzeichnisses mit tausenden Dateien länger dauert als die Anzeige einer handvoll Dateien.

Zudem ermittelt das Programm teilweise komplexe Informationen über die Dateien, es ist also ebenfalls normal, dass die Zeitspanne für den Aufbau der Anzeige über eine bloße Textliste aller Dateien hinausgeht.

Vorgabe: Die Zeitspanne für die Anzeige des gewählten Verzeichnisses soll sich an der Zeitspanne beim Windows Explorer in einer der komplizierteren Ansichten orientieren.

4.Sauberkeit der Operationen

Ein Übertrag des Timestamps von Spalte A nach Spalte B soll genau wie spezifiziert funktionieren, so dass *erstens* nur die markierten Dateien geändert werden und *zweitens* nur die genannte Zielspalte geändert wird und *drittens* danach in der Zielspalte soweit technisch möglich genau der Wert der Quellspalte steht.

5.Benutzerfreundlichkeit

Das Programm soll seine nicht ganz unkomplizierte Funktionalität durch eine saubere Tabelle mit allen Daten und eine Operation mit möglichst wenigen Dialogen und Buttons erreichen.

6.GUI Sprache

Um eine größere Zielgruppe zu erschließen, wird die GUI in englischer Sprache erstellt.

Eine Infrastruktur für die Übersetzung in andere Sprachen ist ein Thema für künftige Versionen, nachdem sich hinreichendes Interesse gezeigt hat.

4. Glossar

Access time (deutsch: Zugriffs-Zeit)

Die Access time notiert den Zeitpunkt des letzten Zugriffs auf eine Datei, also insbesondere auch bei lesenden Zugriffen.

Diese Zugriffszeit ist ein Timestamp von geringerer Relevanz, da er erstens nicht auf allen Betriebssystemen überhaupt mitgeführt wird und da er auch auf MS Windows, wo er in den Filesystemen vorgesehen ist, von Profinutzer typischerweise abgeschaltet wird, um das Spionagepotenzial und die Abnutzung des Speichermediums zu reduzieren.

Control (im GUI Kontext)

Control ist der im Microsoft Umfeld übliche Begriff für ein Widget/Bedienelement

→ siehe Widget

Creation time (deutsch: Erzeugungs-Zeit)

Die Creation time ist prinzipiell der Zeitpunkt der ersten Erzeugung einer Datei.

Bei vielen Betriebssystemen bezieht sich dieser Stamp auf die einzelne Instanz der Datei, im Falle von 3 Kopien des gleichen Inhalts gibt es also 3 verschiedene Creation times, jeweils für das Erstellungsdatum der Kopien.

Das kann zu der merkwürdigen Situation führen, dass eine Datei geändert wurde (**modified**) bevor sie erzeugt wurde (**created**).

Die Creation time hat einen geringen Nutzen und wird von diesem Programm nicht betrachtet.

FAT16/FAT32

→ siehe Filesystem

Filesystem (deutsch: Dateisystem)

Ein Filesystem (Dateisystem) ist ein System zur organisierten Speicherung von Dateien auf einem Datenträger, typischerweise in Verzeichnissen organisiert.

Filesysteme sind für dieses Programm von besonderer Relevanz, da schon auf MS Windows zwei oder drei Filesysteme gebräuchlich sind (FAT32, NTFS, ggf. exFAT), die sich in ihrem Handling von Timestamps erheblich unterscheiden.

Metadaten (für Dateien)

Im Filesystem werden außer dem Dateinamen und dem Dateiinhalt, der meist den größten Umfang hat, noch eine kleine Menge Metadaten über die Dateien abgespeichert.

Diese Speicherung erfolgt nicht bei den Dateiinhalten, sondern eine Strukturebene oberhalb im Verzeichnis.

Diese Metadaten umfassen beispielsweise Dateigröße, verschiedene Timestamps (für Erzeugung, Änderung und Zugriff) sowie Flags, die den Zugriff auf die Datei regeln.

Modification time (deutsch: Änderungs-Zeit)

Die Modification time ist der Zeitpunkt der letzten Änderung des Dateiinhalts und der zentrale Datenpunkt dieses Programms.

NTFS

→ siehe Filesystem

Timestamp (deutsch: Zeitstempel)

Timestamps gehören zu den Metadaten, die im Verzeichnis für jede enthaltene Datei abgespeichert werden. Timestamps beinhalten Datum und Uhrzeit für verschiedene letzte Zugriffe auf diese Datei und werden ohne besonderes Zutun des Nutzers oder der Software bei zahlreichen Operationen aktualisiert.

Typischerweise gibt es die 3 Timestamps
(siehe diese Einzelbegriffe hier im Glossar...):

- creation time
- modification time
- access time (eventuell...)

Widget

Ein Widget (im Microsoft Umfeld „Control“) ist ein einzelnes Bedienelement in der grafischen Benutzeroberfläche eines Programms. Widget ist ein Kunstwort aus Window und Gadget.

Ein Widget kann einfach sein wie ein einzelner Button oder ein Textfeld, aber auch sehr komplex wie eine komplette Tabellenansicht, die schon 80% der Optik einer Tabellenkalkulation wie MS Excel liefert.

Widgets sind quasi der kleine Bruder von Windows (Fenstern);
vollwertige Windows kann der User frei auf dem Bildschirm positionieren,
während Widgets fast immer im Verbund mit anderen auftreten.

Teil 3 - Spezifikationsdokument

1.Datenmodell

Das Datenmodell der Anwendung birgt wenig Neues, da es ja genauer die Aufgabe hat, den in den Systemen vorhandenen Explorer um für Zeitstempel relevante Funktionalität zu erweitern.

Die zentralen Objekte sind also Verzeichnisse, die viele Dateien (und Unterverzeichnisse) enthalten, was eigentlich keiner Visualisierung Bedarf.

Die Besonderheit in diesem Projekt ist, dass die Metadaten über die Dateien über das normale Maß des Explorers hinausgehen - für jede Datei gibt es laut den zuvor genannten Funktionalen Anforderungen 4 Timestamps, weitere werden in künftigen Versionen folgen.



Ob es zudem Daten gibt, die zu jedem Verzeichnis-Objekt gespeichert werden, wird Entwurf ergeben, ggf. werden Informationen über das private Textfile des Programms festgehalten.

2.Kernprozesse

1.ANZEIGE des kompletten Verzeichnisses

mit den speziell hierfür ermittelten und durch Formeln aufbereiteten Metadaten, gegebenenfalls in gewünschter Sortierung

2.WECHSEL des Verzeichnisses auf 3 Wegen

- Pfad-Eingabe in der Adresszeile (Eintippen oder Copy und Paste)
- Navigation in Unterverzeichnisse durch Doppelklick
- Button für Wechsel eine Verzeichnisebene nach oben rechts in der Adresszeile

3.ÜBERTRAG DES TIMESTAMPS

von einer Timestamp Spalte in eine andere;
dies ist der eigentliche *KERNPROZESS* des Programms.

Der Nutzer trifft in der GUI in beliebiger Reihenfolge 3 Entscheidungen:

- Markierung der gewünschten Datei(en) in der Tabellenansicht
- Auswahl einer Quellspalte
- Auswahl der Zielspalte

Wenn alle Auswahlen getroffen sind, kann die Operation gestartet werden.

Die Mächtigkeit und Komplexität dieser Software mit nur EINER KERNOPERATION entsteht durch die gezielte Sortierung und Markierung von Dateien und dann durch die kombinatorische Variation von 4 Quellspalten und 4 Zielspalten (bis zu 12 Varianten, sofern alle Spalten auch als Ziel fungieren).

3. Geschäftsregeln

- nur markierte Dateien werden verändert
- es werden keine Dateiinhalte verändert, nur Timestamps
- wenn die Quellspalte einen Timestamp liefert, wird dieser korrekt in die Zielspalte übertragen (die Genauigkeit kann dabei technischen Einschränkungen unterliegen, falls Quelle und Ziel verschiedene Möglichkeiten haben)
- wenn die Quellspalte keinen Timestamp liefert, was bei bei zwei der vier derzeitigen Spalten möglich ist, bleibt das Ziel unverändert (es wird also nicht mit einem undefinierten Wert ein anderer Timestamp zerstört)

4. Systemschnittstellen

---KEINE---

Natürlich hat das Programm eine Schnittstelle nach unten zum Betriebssystem und Filesystem, aber das gilt ja für nahezu jede Software.

5. Grafische Benutzerschnittstelle (GUI)

Es gibt aktuell nur ein HAUPTFENSTER, es sind keine POPUPS oder MENÜS vorgesehen.

Beschreibung des Hauptfensters:

1. ZEILE OBEN: PFAD DES AKTUELLEN VERZEICHNISSES

- aktueller VERZEICHNIS-PFAD als Text im Format des jeweiligen Betriebssystems
- Schnittstelle zum Explorer via Copy&Paste
- zusätzlich ein [...] oder [...] Button für Navigation eine Verzeichnis-Ebene nach oben

2. MITTE, KERN: TABELLENANSICHT der Verzeichnis-Inhalte

des aktuellen Verzeichnisses mit Dateiname, Größe und vor allem vielen Timestamps

- Markierung von Dateien (Zeilen) als Feature des Listen-Widgets
- Spalten-Überschriften dienen der Sortierung

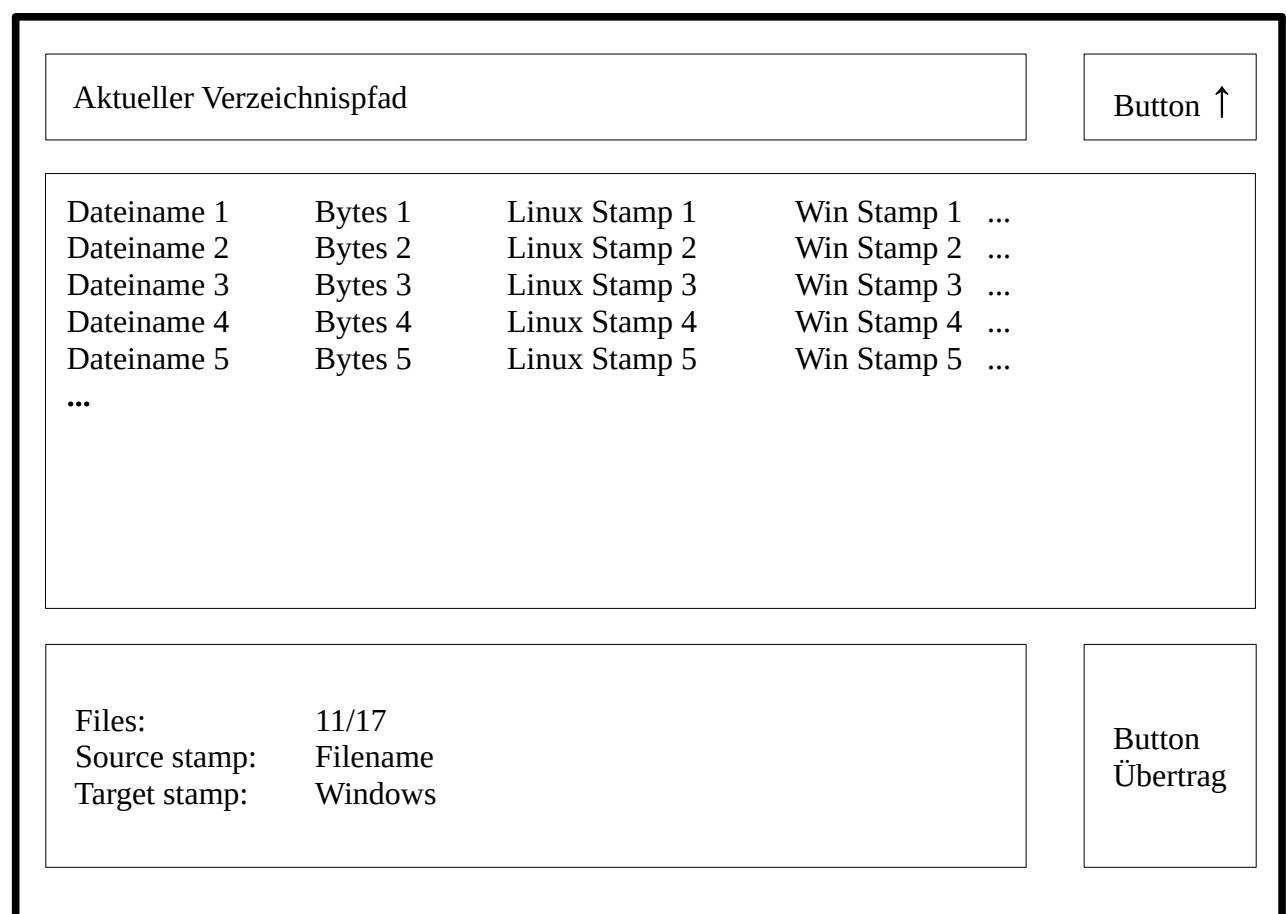
3. ZEILE UNTEN: VORBEREITUNG der ÜBERTRAGS-OPERATION

1.Files: Information über die markierten Dateien

2.Source-Stamp: --- oder Name der Quell-Spalte

3.Target-Stamp: --- oder Name der Ziel-Spalte

sowie ein BUTTON zum Start der Übertrags-Operation



Teil 4 - Architekturdokument und Design

1. Technologieübersicht

Obwohl die Aufgabenstellung für das Modul DLMCSPSE_D lediglich die Implementierung einer Desktopanwendung für Windows verlangt, ist das hier gewählte Tool inhärent Multi-Plattform, also für den Anfang MS Windows und Linux (Entwicklungsplattform ist Debian). Nur wenn es auch auf beiden Plattformen läuft, kann es seine Vorteile wie den sicheren Transport der Stamps im Textformat und die Analyse der Plattform-Unterschiede wirklich ausspielen.

Für diese Multi-Plattform Entwicklung ist die Programmiersprache **PYTHON** die ideale Wahl. Python ist eine Hochsprache mit hervorragenden Eigenschaften und einem ausgereiften Ökosystem.

Aus gleichem Grund ist die Verwendung einer Multi-Plattform GUI Library unabdingbar. Eine Entwicklung für die GUI einer bestimmten Plattform (auf Windows früher Win32, heute .Net, auf Linux Gnome oder KDE) ist heute ohnehin nicht mehr zeitgemäß, ganz sicher aber nicht für ein derartiges Multi-Plattform Tool.

Der Bedarf nach einem guten Tabellen Widget (siehe Risikoanalyse, Risiko No.1) erfordert zudem eine große, ausgereifte GUI Library, so dass von vorn herein nur wxWidgets (wxPython) und Qt for Python infrage kamen.

Dabei ist Qt allerdings ein kommerzielles Produkt mit einer ungünstigeren Lizenz und erscheint zudem in der Python-Bindung umständlicher (Qt ist deutlich als dünner Wrapper um C++ erkennbar), so dass Qt nur zum Einsatz gekommen wäre, wenn das Tabellen-Control in wxWidgets unzureichend gewesen wäre.

Die Befürchtung eines schlechteren Tabellen-Controls in wxWidgets (das Risiko 1) stammte aus Erfahrungen in einem früheren Projekt, wo das Html-Control von wx tatsächlich unzureichend war und Qt eingesetzt werden musste. Beim Tabellen-Control (wx.grid.Grid ganz konkret) war diese Befürchtung aber unbegründet, es passt perfekt für das Projekt.

Als GUI Library kommt somit wxWidgets bzw. genaugenommen **wxPython** zum Einsatz. Siehe <https://wxpython.org/> als Einstieg und Dokumentation; die Installation sollte wie üblich mit "pip install wx" erfolgen.

2. Architekturübersicht

1. GUI PROGRAMM folgt den Vorgaben von wxWidgets

Für die oberen Klassen der Architektur ist keine lange Begründung erforderlich, als GUI Programm auf Basis der GUI Library wxWidgets folgt diese Schicht den Vorgaben dieser Library.

Das betrifft im Hauptprogramm "timestamper_gui.py" den __main__ Zweig und die beiden Toplevel Klassen, die unten beschrieben werden.

2. MODULARITÄT, Trennung von basis.py und timestamper_gui.py

Die Klassen in basis.py werden so angelegt, dass sie die puren Timestamp-Berechnungen und weiteren File- und Verzeichnis-Eigenschaften ohne jede Bindung an die GUI enthalten (insbesondere erfolgt kein Import von wxWidgets).

Diese Art von Trennung von Library und aufgesetzter GUI (bzw. Business-Logik und Präsentationsschicht) ist weit verbreitet. Auf diese Weise sollen die Kernfunktionen des Programms auch ganz ohne bunte GUI verfügbar gemacht werden.

Für dieses Projekt (aber nicht mehr für die Abgabe in diesem Modul) wird zeitnah eine Kommandozeilen-Version "timestamper_cmd.py" entstehen, die Kernfunktionen auf Plattformen anbietet, wo man wxWidgets nicht installieren kann (z.B. im Termux Linux Emulator auf dem Android Handy, oder auf altem Windows 98, etc.) oder aufgrund des Umfangs nicht will. Anstelle der Anzeige in der GUI tritt dann ein ls/dir artiges Listing mit allen Timestamps nach stdout und Operationen wie TRANSFER müssten über die Kommandozeile angestoßen werden.

Wichtig ist aber, dass die Kernfunktionen von CellObjects auch hier direkt anwendbar sein werden. Man braucht auch wieder eine Stringform .getDisplay() und eine Sortierung .getSortKey().

Aus Effizienzgründen gibt es eine kleine Besonderheit. Um zu vermeiden, dass Farben für die Anzeige in wx tausende Male immer wieder aus Config-Tupeln konvertiert werden, speichert die Basis fertige Farbwerte.

Das passiert aber sauber und ohne Zerstörung der Isolation der Module, indem die GUI der Basis eine Konvertierungsfunktion zur Verfügung stellt und die Basis dann für sie opaque ("black box") Farbwerte abspeichert. Eine Kommandozeilenversion ohne Bedarf an Farben kann dann einfach "lambda t: None" als Farbfunktion abliefern.

Das basis Modul ist übrigens letztlich um den Faktor 1.5 größer als das GUI Modul, was ein guter Zustand ist.

3. Objektorientierung, POLYMORPHISMUS

Ansonsten ist noch die intensive Verwendung von Objektorientierung, insbesondere Polymorphismus für die low level Datenklassen in der Basis erwähnenswert.

Das Hauptprogramm trifft nur 2 Annahmen über Zellen-Objekte, verwendet also nur 2 Interfaces: CellObject im Allgemeinen und das erweiterte CellObjectStamp in Spalten, die bekanntermaßen Timestamps enthalten.

Diese beiden Basisklassen, die ganz oder teilweise abstract sind, werden dann durch derzeit 15 Unterklassen implementiert. Die genauere Beschreibung folgt im nächsten Abschnitt.

3. Struktur, Klassen

1. GUI KLASSEN

Die Klassen in der GUI Schicht sind wie gesagt in ihrer Struktur durch die Verwendung der GUI Library wxWidgets vorgegeben.

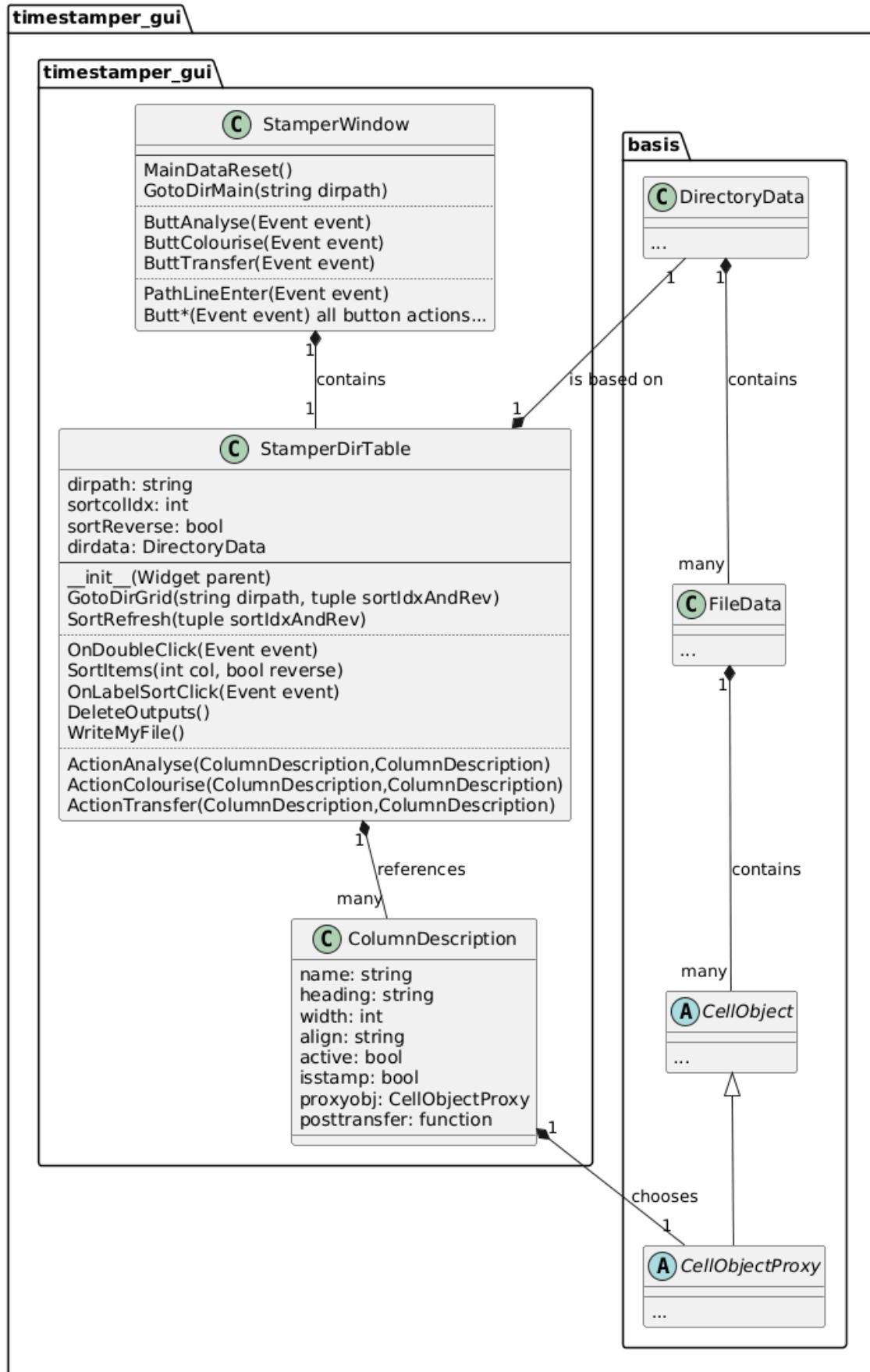
Das HAUPTFENSTER ist eine Instanz von StamperWindow, abgeleitet von wx.Frame, das KERNELEMENT File-Tabelle ist eine StamperDirTable, abgeleitet von wx.grid.Grid.

Im **HAUPTFENSTER** erfolgt im Konstruktor der Aufbau der gesamten GUI und als Event-Callbacks gibt es vor allem die Button-Reaktionen der ganzen über die GUI verteilten Buttons. Im Hauptfenster finden sich Events, Methoden und Daten, die über die Tabelle selbst hinausgehen und außerhalb der Tabelle in weiteren Widgets angezeigt werden.

Der KERN des Programms ist die **TABELLE** StamperDirTable. Hier erfolgt die eigentliche Anzeige der Tabelle mit Farben und Formatierungen. Die Event-Reaktionen sind hier tabellen-spezifisch wie verschiedenartige Klicks an verschiedenen Orten der Tabelle.

Die komplette Beschreibung der Optik der großen Tabelle erfolgt nicht über Code sondern über Daten in Form von ColumnDescription Objekten.

Diese enthalten neben simplen Werten wie Headlines und einfachen Formatierungen sowie Flags für Timestamps auch CellObjectProxy Objekte, die die primitive Inhaltswerte für ihre Verwendung in der Tabelle aufbereiten.



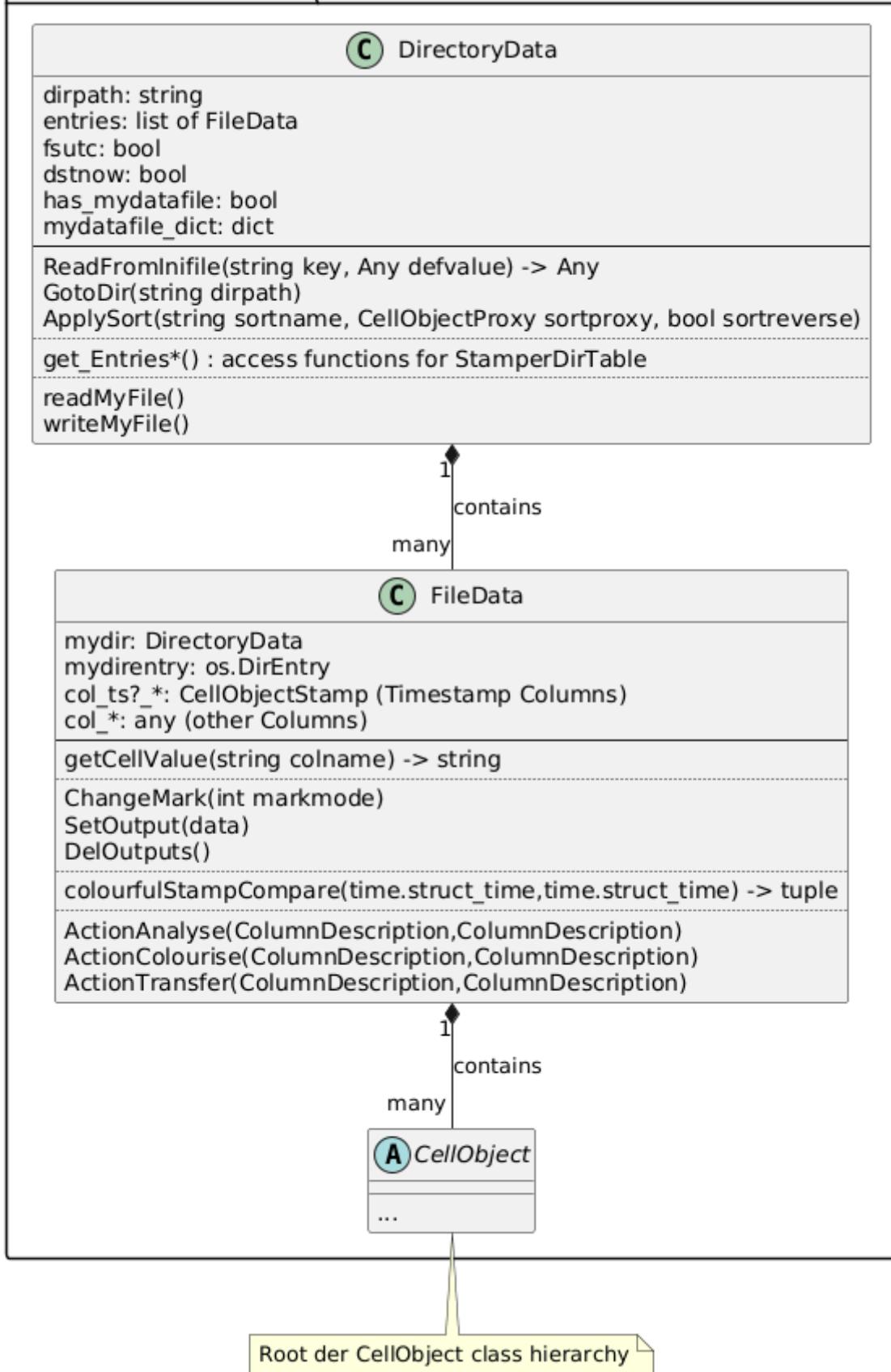
2.TOP-LEVEL KLASSEN DER BASIS

Die 2 Toplevel Klassen in der Basis sind DirectoryData für das aktuelle Verzeichnis und FileData für die Files/Entries darin.

Zwischen **DirectoryData**, dem Verzeichnisobjekt in der Basis und StamperDirTable in der GUI gibt es eine 1:1 Beziehung, das DirectoryData ist also die direkte Entsprechung der Table in der Basis, ein FileData Objekt entspricht dann einer Zeile darin.

Für die Sprechweise ist zu beachten, dass mit DirectoryData das aktuell angezeigte Verzeichnis gemeint ist. Subdirectories darin sind keine DirectoryData Objekte sondern FileData mit dem .getType 'D' und reduzierten Timestamp Eigenschaften.

Die Fileobjekte **FileData** haben member Variablen für jede Zelle in ihrer Tabellenzeile. Einfache Spalten speichern dabei einfache Datentypen wie int oder string, komplizierte Spalten sind **CellObject** Objekte, wobei Timestamps Unterklassen von CellObjectStamp sind.

timstamper_gui.basis

3. CellObject DREI KLASSENBÄUME

Auf dem untersten Level steht für jede "komplizierte" Zelle in der Zeile einer Datei ein CellObject. Durch die pure Verwendung des Worts "Zelle" entsteht noch keine Bindung an die GUI, denn auch andere Hauptprogramme die die Basis nutzen werden, werden eine Art Tabellen-Metapher verwenden.

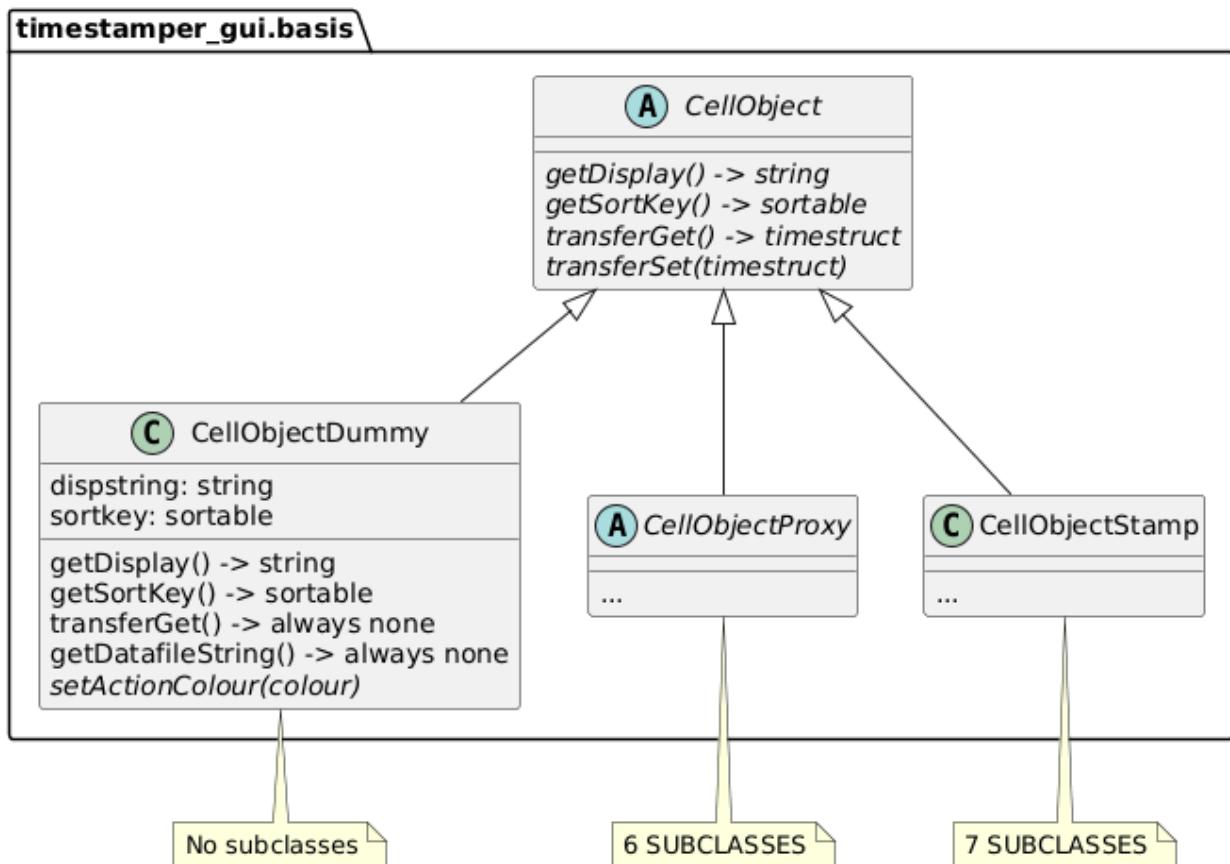
In den Zellen der Fileobjekte braucht es aber eine Trennung von einfachen Datentypen (int, string,...) für einfache Spalten und echten CellObject Objekten.

Auf der einen Seite sind für Timestamps mit ihrer komplizierten Logik zwingend vollwertige Objekte erforderlich, auf der anderen Seite soll für 3000 Files nicht 3000 mal der einfache Integer "Bytecount" als Objekt verpackt werden.

Das ist die Motivation für Proxies.

CellObjectProxy werden oben in der GUI in der ColumnDescription genannt und übernehmen Aufbereitung von einfachen Datentypen (und None für leere Zellen) für diese Spalte. Diese Bindung wurde auf dem 1.UML Klassendiagramm angedeutet.

Die "echten" CellObject, die in den Filedata gespeichert werden, sind derzeit alles Nachfahren von **CellObjectStamp**, also Zeitstempel. Die Beschreibung folgt.



CellObjectDummy sind ein Sonderfall, der zwischen vollen CellObject und einem primitiven Typ mit Verpackung als CellObjectProxy steht. Hier wird zwar unten im Fileobjekt ein Objekt für die Zelle gespeichert und kein primitiver Datentyp, dieses ist aber kein vollwertiges Objekt, sondern es wiederholt auf Anfrage nur die genannten Werte.

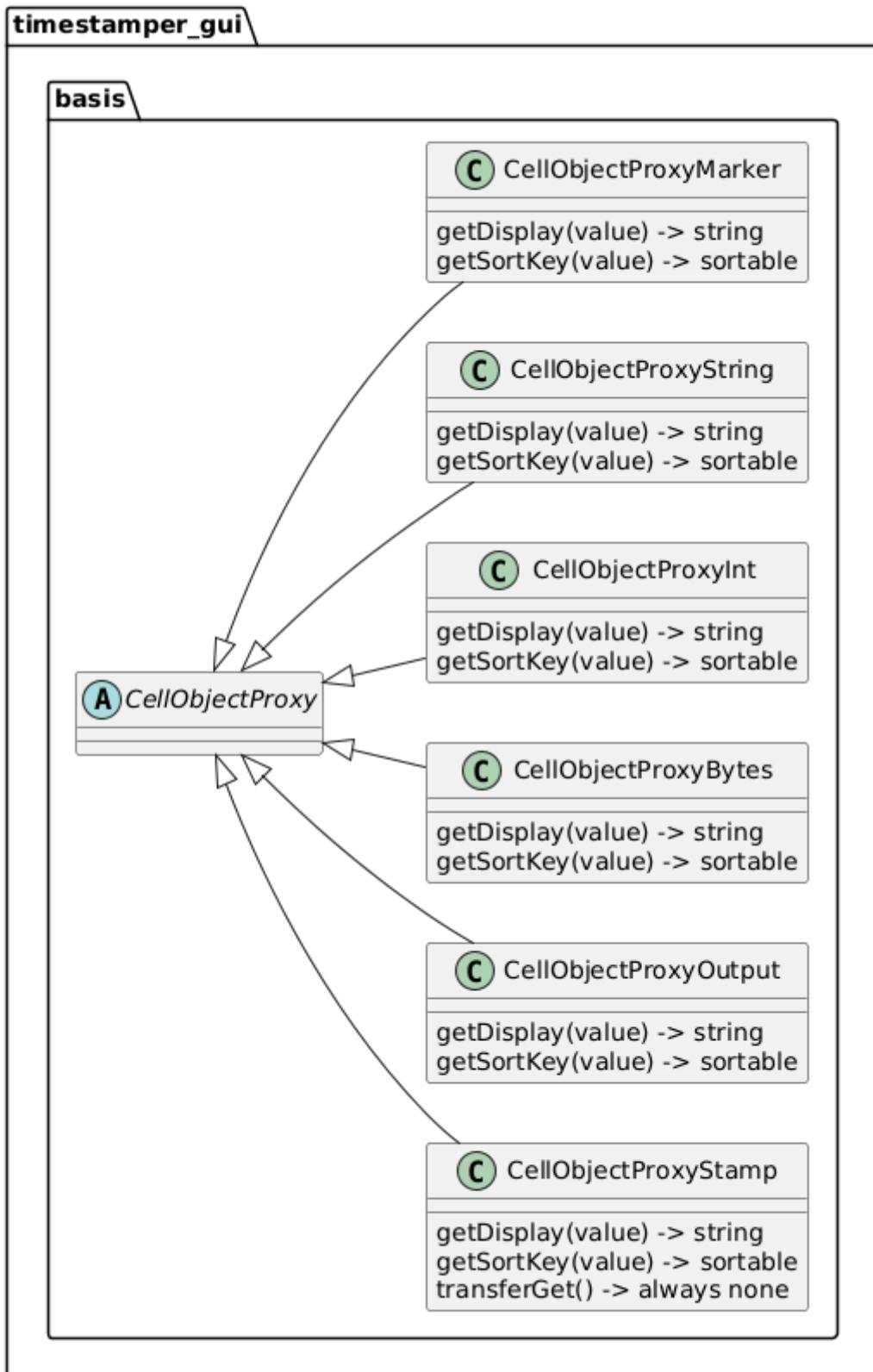
Einige Verwendung ist derzeit, um in der Sortierung von Timestamps Verzeichnisse, die ohnehin nie Timestamps anzeigen, von Files, die nur an dieser konkreten Stelle keinen Stamp haben, zu trennen.

4. CellObjectProxy als Wrapper für simple Typen

Proxy Klassen gibt es in reicher Auswahl, das soll aber nicht darüber hinwiegäuschen, dass die durchschnittliche Größe einer CellObjectProxy Implementierung im Code letztlich nur 10 Zeilen groß ist.

Das ist alles was es braucht, um je nach Spalte die primitiven Daten für die Anzeige (getDisplay) und Sortierung (getSortKey) aufzubereiten, wie das jede CellObject Klasse zu tun hat.

Zu beachten ist hier, dass die Proxies ihren Datenwert für diese Funktionen genannt bekommen müssen, weil sie eben Proxies dafür sind und nur einmal pro Spaltenbeschreibung instanziiert werden.



5. CellObjectStamp die eigentlichen Timestamps

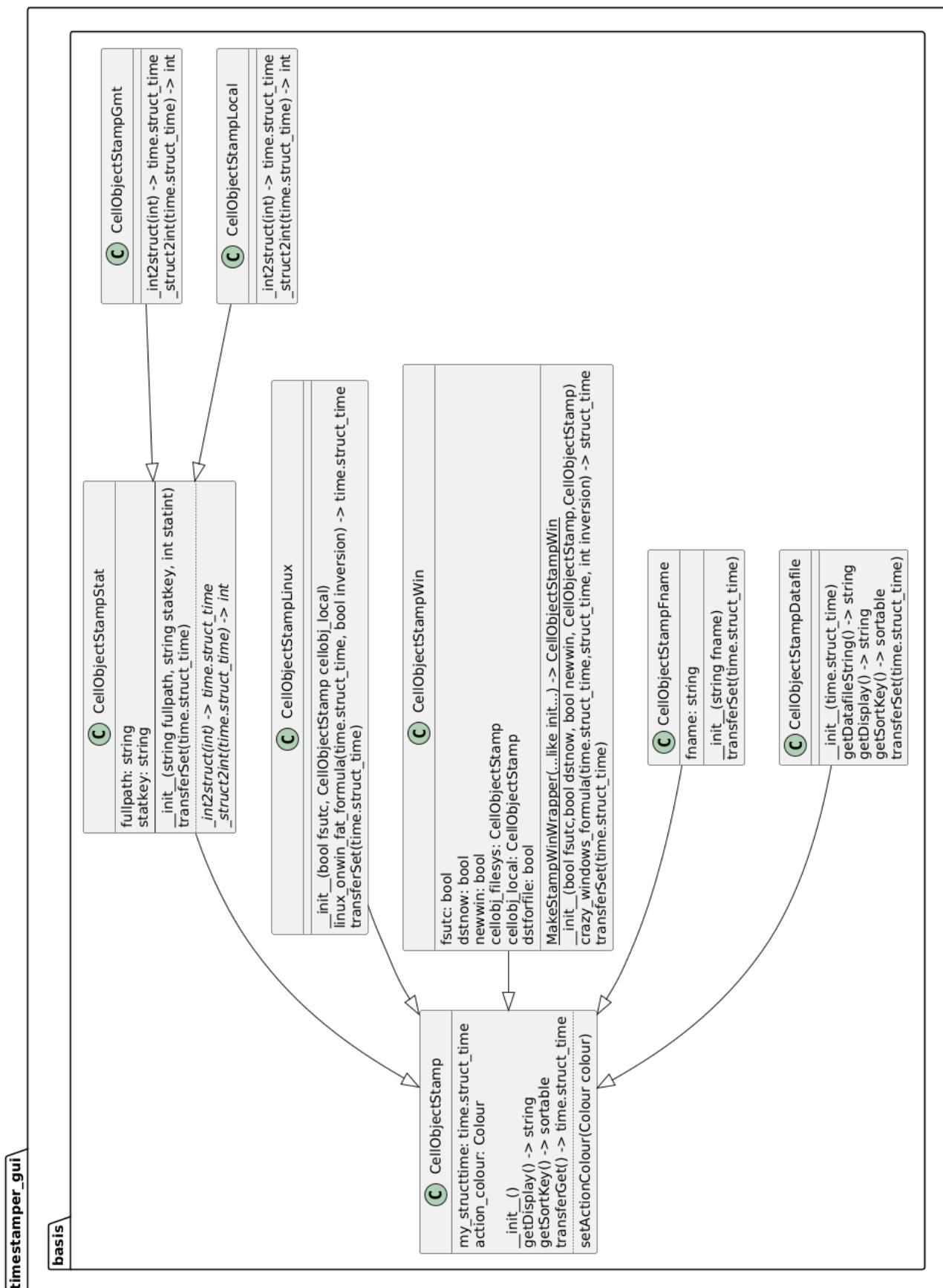
Die CellObjectStamp Nachfahren sind schließlich der eigentlich KERN des ganzen Programms.

Es gibt 7 Unterklassen, die in der GUI für die Anzeige von ebensovielen Timestamp-Spalten verwendet werden (wobei die Abbildung nicht genau 1:1 ist).

Am besten betrachtet man **CellObjectStamp...** Klassen in 4 Gruppen:

- **...Stat** sowie die Unterklassen **...Local** und **...Gmt**: haben eine direkte Bindung an die Zeitstempel im Filesystem
- **...Linux** und **...Win** sind Wrapper um die eben genannten Klassen, um per Formel die "Meinung" auf verschiedenen Plattformen abzubilden
- die **...Fname** sind eine reine Datenquelle, hier kommt die Zeitinformation [soweit vorhanden] aus dem Filename
- **...Datafile** stellt schließlich die Bindung an das eigene, private Datenfile mit gespeicherten Stamps her

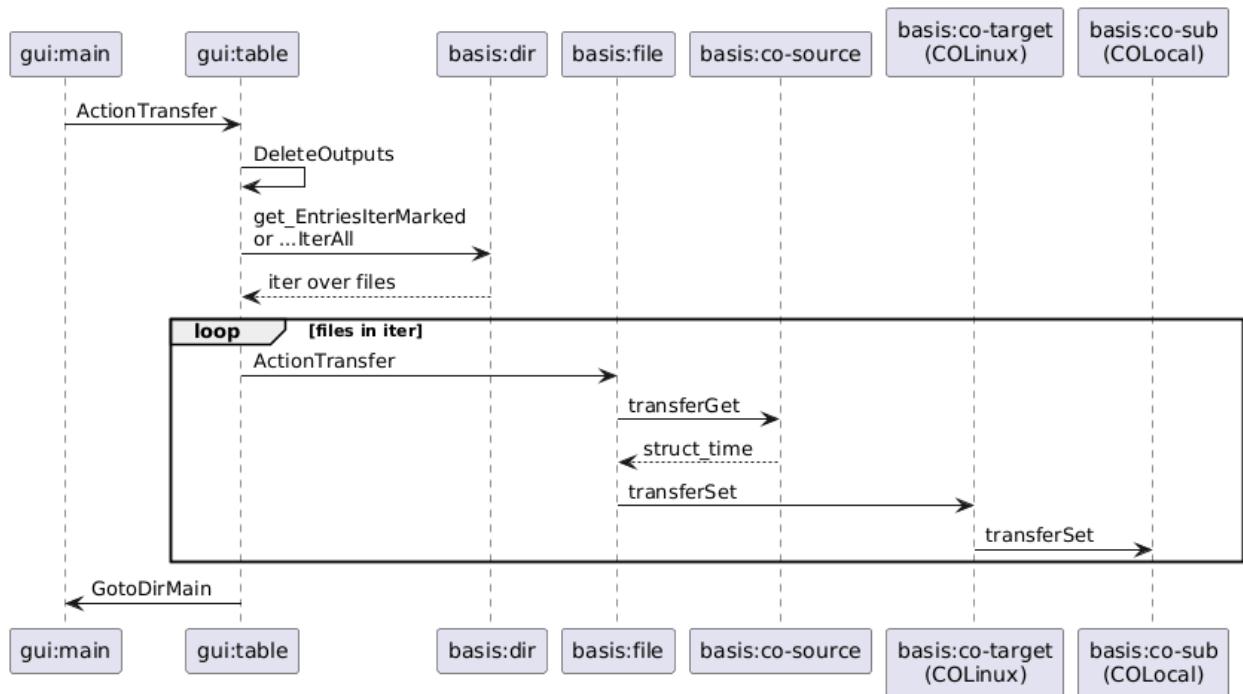
Die Idee daran ist, wie in der User Doku später erklärt wird, dass man an einem Ort/auf einer Plattform, wo die Timestamps noch korrekt sind, diese in diesem File "einfriert" und von da an vor den Unwägbarkeiten der Systeme sicher ist.



4.Verhalten

Als zentraler Systemablauf soll die Operation TRANSFER beschrieben werden, weil sie von Anfang an als Kern des Programms spezifiziert war und weil sie auch einige Besonderheiten hat, für die eine Erklärung lohnt.

Die Akteure im UML Sequenzdiagramm sind allesamt Objekte der Programmlogik, zuerst die 2 großen Widgets aus der GUI, dann die 2 Top Objekte der Basis und schließlich ganz unten die CellObject Instanzen. Der User tritt nur mit einem einzigen Knopfdruck ganz am Anfang in Erscheinung, was nicht dargestellt ist.



Das **ERSTE SEQUENZDIAGRAMM** zeigt den Transfer aus einer beliebigen anderen Timestamp-Spalte in die Spalte "Linux", also den Stamp wie Linux ihn sieht/sehen würde. Ein Klick auf den [TRANSFER] Button im Hauptfenster (StamperWindow) startet aus dem Button-Callback die ActionTransfer Aktion der großen Tabelle (StamperDirTable), die der Hauptakteur für diesen Ablauf ist.

Die Tabelle löscht zuerst mit DeleteOutputs eventuelle farbige Outputs in ihren Zellen, damit sich nicht die Outputs mehrerer Operationen überlagern.

Die Tabelle fragt dann in der Basis bei ihrem verlinkten Directory-Objekt (DirectoryData) nach einem Iterator entweder über alle Files (get_EntriesIterAll) oder über die markierten Files (get_EntriesIterMarked).

In einer Schleife über alle diese genannten Files (FileData) ruft die Tabelle jeweils ActionTransfer für das File und nennt dabei source und target Spalte. Die Tabelle hat dabei nur minimales Wissen über die Struktur von FileData Objekten, sie setzt allerdings voraus, dass für

die 3 ActionXXXXX Operationen, die sie selber implementiert, auch unten ActionXXXXX Operationen existieren.

Das jeweilige File Objekt (FileData) hantiert mit zwei CellObjectStamp Objekten, der Quelle und dem Ziel des Transfers.

Es fragt die Quelle mit transferGet nach den Daten für den Transfer und erhält die relevanten Daten (konkret einen time.struct_time Record, aber das braucht das Fileobjekt nicht zu interessieren) und gibt sie mit transferSet an das Ziel weiter.

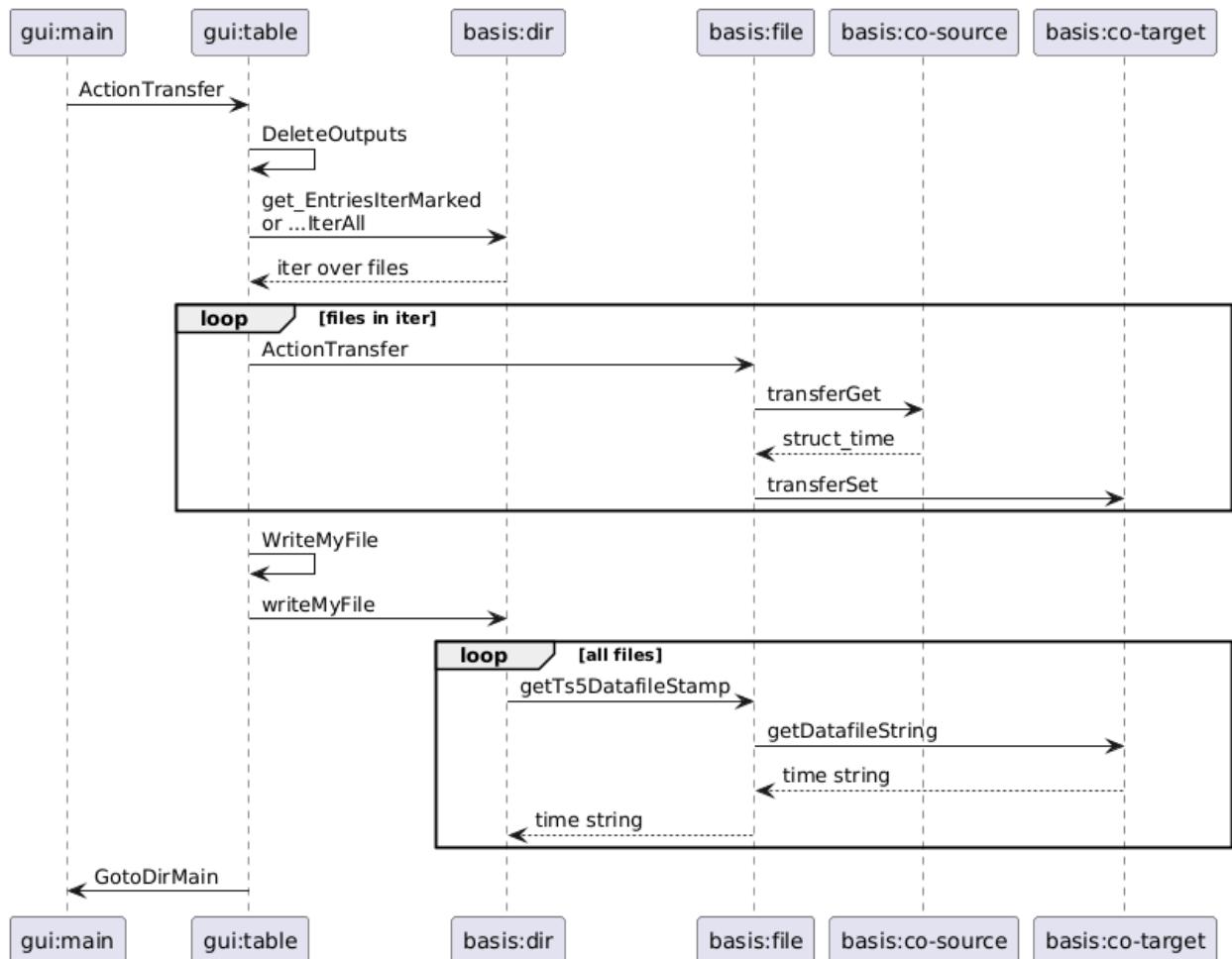
Hier im ersten Sequenzdiagramm erfolgt der "unterste" Schritt und die eigentliche Veränderung des Timestamps im Filesystem, wenn die Wrapperklasse CellObjectStampLinux die nur ihr bekannte Anpassungsformel auf den Wert anwendet und dann einen neuen Wert an ihr Basis-Objekt CellObjectStampLocal erneut per transferSet weiterreicht.

Nach Ende der Schleife endet die Operation hier ohne Post-Processing, indem die Tabelle das Hauptfenster mit GotoDirMain zum Update auffordert.

In einem **ZWEITEN SEQUENZDIAGRAMM** soll eine Variation des Transfers gezeigt werden, wo ein Post-Processing nach der Schleife über alle Files erforderlich ist, um einmal nach dem kompletten Update die neuen Stamps aller Files in das private Datenfile des Programms zu schreiben.

Der gesamte Ablauf bis zum ersten transferSet an das Zielobjekt (hier ein CellObjectStampDatafile) ist komplett gleich, nur dass dieses Objekt kein weiteres Unterobjekt hat, sondern den Wert selbst speichert.

Danach - nachdem alle Fileobjekte verarbeitet sind - erfolgt durch die Table ein zusätzlicher Postprocessing-Schritt.



Dieser ist in der Sequenz als einfacher Aufruf von WriteMyFile durch die Tabelle an sich selbst dargestellt, tatsächlich ist der Ablauf etwas komplizierter, da auch die Notwendigkeit für Postprocessing wiederum komplett über Daten beschrieben ist.

(Die Tabelle findet in der ColumnDescription der Zielspalte kein einfaches "JA" für die Verwendung als Transferziel sondern den Callback für eine "posttransfer" Funktion, quasi ein "JA, ABER". Diese Funktion ist in diesem (und derzeit einzigen) Falle die Funktion DirWriteMyFile und erst diese Funktion ruft für die Tabelle WriteMyFile)

Die Tabelle gibt diese Anweisung einfach an ihr Directory-Objekt als Aufruf writeMyFile weiter; das Schreiben von Files ist keine Aufgabe der GUI Schicht.

Das Directory-Objekt läuft wieder über die Files (diesmal alle Files) und fragt sie mit getTs5DatafileStamp nach ihrem Timestamp in Textform für das File. Die Files leiten diese Frage mit getDatafileString an das richtige CellObject weiter.

Wenn das Directory-Objekt alle File-Zeilen geschrieben und das private Datenfile geschlossen hat, ist das Postprocessing beendet und die ActionTransfer endet wie gehabt mit der Aufforderung zum Update an das Hauptfenster mit GotoDirMain.

5. Erweiterung der Spezifikation um zusätzliche Funktionalität

Erfreulicherweise konnte die Spezifikation vollständig umgesetzt werden.

Alle Änderungen der Spezifikation in dieser Phase 2 sind daher Erweiterungen um neue Funktionen, deren Notwendigkeit sich im Laufe der Entwicklung ergeben hat.

1. erweiterte MARKIERUNG VON FILES für Operationen

Die ursprüngliche Spezifikation legte nur fest, dass die Markierung von Files für die Operationen mit Mitteln der GUI Library erfolgen soll:

"für die Bedienung der Markierung und die visuelle Darstellung kann man sich hoffentlich auf den Support der GUI Library verlassen"

Es hat sich bei der Entwicklung ergeben, dass die bloße Nutzung der Selection des Tabellen-Widgets (welche Zeilen sind gerade angeklickt) unzureichend ist. Diese Selection ist zu flüchtig und zu schnell versehentlich geändert, um sie für potentiell gefährliche Operationen zu verwenden.

Die finale Lösung ist also eine Kombination aus Selection im Tabellen-Widget und anschließendem **Übertrag** in eine extra Markierungs-Spalte, wodurch sich ein mächtiges Werkzeug zur ganz gezielten Auswahl von Files nach ganz bestimmten, heterogenen Kriterien ergibt.

2. NEUE SPALTEN

Die ausführlichen Analysen für die Erstellung der richtigen Timestamp-Formeln ergab zahlreiche zusätzliche "Debugging" Spalten, die aktuell teilweise noch angezeigt werden, aber leicht an- und abgeschaltet werden können.

Zudem entstand durch die mögliche Abweichung zwischen Python und Linux (konkret "Was würde Linux anzeigen" für ein FAT Laufwerk wenn das Programm aktuell auf Windows läuft) die Notwendigkeit einer weiteren "offiziellen" Ausgabespalte, die Hauptspalten sind also nunmehr diese fünf:

- Linux (GEÄNDERT, anstatt Python)
- Windows alt...
- ... und neu
- aus Filename
- im privaten Datenfile

3.DREI STATT NUR EINER KERNFUNKTION

Beim Entwickeln der recht komplizierten Zeitanpassungen, um auf einem System (Linux) die Meinung eines anderen Systems (old Windows) von den Stamps dieser Dateien anzuzeigen, war es immer wieder nötig, Zeitstempel für lange Listen von Dateien "mit dem Auge" zu vergleichen.

So entstand schnell der Wunsch nach eingebauten Analyse-Funktionen, die letztlich sogar noch vor der eigentlich geplanten TRANSFER Funktion entwickelt wurden.

1.neue Funktion ANALYSE

vergleicht Zeitstempel in 2 Spalten und zeigt die Differenz in Sekunden, bildet aber vor allem eine ABSTRAKTION "wie schlimm" der Unterschied ist, was mit einem Textkürzel und einem Ampel-artigen Farocode angezeigt wird.

[python-ide] TimeStamper GUI															
/home/user/Modul4-Bilder2/Modul4 BeispielFotosZeitgrenze (via exFat=OK)															
[X] [I] [/]															
Mark	Typ	Name	tensi	Bytes	***	PY local	PY gmt	DST	Linux	Win new	Win old	From	Fname	My Datafile	
1	F	timestamp-data.txt	.txt	742											
2	F	rs10-250810-150341 Sommer.jpg	.jpg	2.106.178	1:fat:1	15:03:42	13:03:42		1 15:03:42 10 15:03:42 .0 14:03:42	2025-08-10 15:03:41	2025-08-10 15:03:42				
3	F	rx104-08363 Sommer.jpg	.jpg	5.691.476		15:20:24	13:20:24		1 15:20:24 10 15:20:24 .0 14:20:24					2025-08-10 15:20:24	
4	F	rs10-250812-130452 Sommer.jpg	.jpg	4.147.544	0:equal:0	13:04:52	11:04:52		1 13:04:52 12 13:04:52 .2 12:04:52	2025-08-12 13:04:52	2025-08-12 13:04:52				
5	F	rx104-08516 Sommer.jpg	.jpg	7.469.416		19:30:28	17:30:28		1 19:30:28 12 19:30:28 .2 18:30:28					2025-08-12 19:30:28	
6	F	rx104-08616 Sommer.jpg	.jpg	6.561.920		16:26:36	14:26:36		1 16:26:36 13 16:26:36 .3 15:26:36					2025-08-13 16:26:36	
7	F	s20-251022-162822 Sommer.jpg	.jpg	2.895.169	0:equal:0	16:28:22	14:28:22		1 16:28:22 2 16:28:22 2 15:28:22	2025-10-22 16:28:22	2025-10-22 16:28:22				
8	F	s20-251022-164005 Sommer.jpg	.jpg	3.475.053	1:fat:1	16:40:06	14:40:06		1 16:40:06 2 16:40:06 2 15:40:06	2025-10-22 16:40:05	2025-10-22 16:40:06				
9	F	s20-251024-135713 Sommer.jpg	.jpg	2.554.399	1:fat:1	13:57:14	11:57:14		1 13:57:14 14 13:57:14 14 12:57:14	2025-10-24 13:57:13	2025-10-24 13:57:14				
10	F	rs10-251024-154034 Sommer.jpg	.jpg	2.996.606	1:fat:2	15:40:36	13:40:36		1 15:40:36 14 15:40:36 14 14:40:36	2025-10-24 15:40:34	2025-10-24 15:40:36				
11	F	s20-251026-155650 Winter.jpg	.jpg	4.141.616	1:fat:2	15:56:52	14:56:52		0 15:56:52 2 16:56:52 2 16:56:52	2025-10-26 15:56:50	2025-10-26 15:56:52				
12	F	s20-251027-132000 Winter.jpg	.jpg	4.336.386	0:equal:0	13:20:00	12:20:00		0 13:20:00 2 13:20:00 2 13:20:00	2025-10-27 13:20:00	2025-10-27 13:20:00				
13	F	s20-251028-135505 Winter.jpg	.jpg	3.473.903	1:fat:1	13:55:06	12:55:06		0 13:55:06 2 13:55:06 2 13:55:06	2025-10-28 13:55:05	2025-10-28 13:55:06				
14	F	rx104-08739 Winter.jpg	.jpg	3.878.488		00:03:46	23:03:46		0 00:03:46 1 00:03:46 1 00:03:46					2026-01-01 00:03:46	
15	F	rx104-08752 Winter.jpg	.jpg	3.980.765		00:06:32	23:06:32		0 00:06:32 1 00:06:32 1 00:06:32					2026-01-01 00:06:32	
16	F	rx104-08756 Winter.jpg	.jpg	3.574.231		00:07:42	23:07:42		0 00:07:42 1 00:07:42 1 00:07:42					2026-01-01 00:07:42	

Sort: Linux (A..Z)

both from-> marked: 0 files/0 items of 16 total
from: My Datafile
to: From Frame

swap ->to

Analysen Buttons: Refresh, Go Up, win=0 utc=1, [A..Z], [Z..A], Colourise, Transfer!

2. neue Funktion COLOURISE

zeigt nur den Ampel-Farbcodes (den man nach mehrmaliger Anwendung von Analyse leicht versteht), dafür aber auf Wunsch in allen anderen Timestamp-Spalten

(python-ide) TimeStamper GUI																
/home/user/Modul4-Bilder2/Modul4 BeispieleFotosZeitgrenze (via exFat=OK)																
[X]	[I]	[U]													Refresh	Go Up
Mark	Typ	Name	tensi	Bytes	***	PY local	PY gmt	DST	Linux	Win new	Win old	From Fname	My Datafile			
1	F	timestamp-data.txt	.txt	742		15:03:42	13:03:42	1	2025-08-10 15:03:42	0 15:03:42	0 14:03:42	2025-08-10 15:03:41	2025-08-10 15:03:42			
2	F	rs10-250810-150341 Sommer.jpg	.jpg	2.106.178		15:20:24	13:20:24	1	2025-08-10 15:20:24	0 15:20:24	0 14:20:24		2025-08-10 15:20:24			
3	F	rx104-08363 Sommer.jpg	.jpg	5.691.476		13:04:52	11:04:52	1	2025-08-12 13:04:52	0 13:04:52	0 12:04:52	2025-08-12 13:04:52				
4	F	rs10-250812-130452 Sommer.jpg	.jpg	4.147.544		19:30:28	17:30:28	1	2025-08-12 19:30:28	0 19:30:28	0 18:30:28		2025-08-12 19:30:28			
5	F	rx104-08516 Sommer.jpg	.jpg	7.469.416		16:26:36	14:26:36	1	2025-08-13 16:26:36	0 16:26:36	0 15:26:36		2025-08-13 16:26:36			
6	F	rx104-08616 Sommer.jpg	.jpg	6.561.920		16:28:22	14:28:22	1	2025-10-22 16:28:22	0 16:28:22	0 15:28:22	2025-10-22 16:28:22				
7	F	s20-251022-162822 Sommer.jpg	.jpg	2.895.169		16:40:06	14:40:06	1	2025-10-22 16:40:06	0 16:40:06	0 15:40:06	2025-10-22 16:40:05	2025-10-22 16:40:06			
8	F	s20-251022-164005 Sommer.jpg	.jpg	3.475.053		13:57:14	11:57:14	1	2025-10-24 13:57:14	0 13:57:14	0 12:57:14	2025-10-24 13:57:13				
9	F	s20-251024-135713 Sommer.jpg	.jpg	2.554.399		15:40:36	13:40:36	1	2025-10-24 15:40:36	0 15:40:36	0 14:40:36	2025-10-24 15:40:34	2025-10-24 15:40:36			
10	F	rs10-251024-154034 Sommer.jpg	.jpg	2.996.606		15:56:52	14:56:52	0	2025-10-26 15:56:52	0 15:56:52	0 15:56:52	2025-10-26 15:56:50	2025-10-26 15:56:52			
11	F	s20-251026-155650 Winter.jpg	.jpg	4.141.616		13:20:00	12:20:00	0	2025-10-27 13:20:00	0 13:20:00	0 13:20:00	2025-10-27 13:20:00				
12	F	s20-251027-132000 Winter.jpg	.jpg	4.336.386		13:55:06	12:55:06	0	2025-10-28 13:55:06	0 13:55:06	0 13:55:06	2025-10-28 13:55:05	2025-10-28 13:55:06			
13	F	s20-251028-135505 Winter.jpg	.jpg	3.473.903		00:03:46	23:03:46	0	2026-01-01 00:03:46	0 00:03:46	0 00:03:46	2026-01-01 00:03:46				
14	F	rx104-08739 Winter.jpg	.jpg	3.878.488		00:06:32	23:06:32	0	2026-01-01 00:06:32	0 00:06:32	0 00:06:32	2026-01-01 00:06:32				
15	F	rx104-08752 Winter.jpg	.jpg	3.980.765		00:07:42	23:07:42	0	2026-01-01 00:07:42	0 00:07:42	0 00:07:42	2026-01-01 00:07:42				
16	F	rx104-08756 Winter.jpg	.jpg	3.574.231												
both	from->	marked:	0 files/0 items of 16 total												?	X
swap	->to	from:	My Datafile												Analysse	X
		to:	---												Colourise	X
															Transfer!	X
Sort: Linux (A..Z)																

Nochmal ein Hinweis zur Nutzung von Englisch im Programm.

Nachdem die Aufgabenstellung dazu keine Vorgabe macht, wurde das Programm wie anfangs spezifiziert zur besseren Wiederverwendung auf Englisch entwickelt.

Das betrifft nunmehr nicht nur die GUI (wo es ohnehin nur um ganze 9 Worte geht!) sondern auch Bezeichner und Kommentare im Code.