Silesian University of Technology

Binary file handling tool

Software Design

AUTHOR: Anna Brzezina, Katarzyna Chowańska, Karolina Wylężek

E-MAIL: annabrz549@student.polsl.pl, k.m.chowanska@gmail.com, karolina.v.wylezek@gmail.com

CREATED: 17^{th} November 2020

MODIFIED: 20^{th} November 2020

RECIPIENTS: Jakub Nalepa, BEng, PhD

VERSION: [1.0.0]

CLASSIFICATION: University confidential

Table of Contents

1. Document History

Version	Date	Author	Approved by	Additional information
0.1.0	2020/11/17	Anna Brzezina, Katarzyna Chowańska, Karolina Wylężek	Anna Brzezina, Katarzyna Chowańska	creating the document, creating the division of the document, adding first version of diagrams
0.2.0	2020/11/18	Anna Brzezina	Anna Brzezina, Katarzyna Chowańska	improving the logical division of the document
0.3.0	2020/11/19	Karolina Wylężek, Katarzyna Chowańska	Anna Brzezina, Katarzyna Chowańska	updating diagrams and adding class descriptions
1.0.0	2020/11/20	Anna Brzezina	–	release candidate

2. Introduction

This document presents the software design established during a CRC session which took place on November 16, 2020. It includes the internal system architecture being a result of the meeting and the UML diagrams — a class diagram and a sequence diagram.

2.1. Definitions, Acronyms and Abbreviations

The following notions will be used in this document:

Term	Explanation	Synonyms
Actor	Any person using the produced software for their or their own business purposes.	user, person
Batch processing	Processing of several input files in a parallel manner (at the same time).	parallel processing
Binary file	A file stored in binary format	input file
CRC	Class-Responsibility-Collaboration — a method helpful in identifying the required classes; on the top of the paper the name of the class is written, on the left of the paper — the responsibilities/duties of the class and on the left — the classes that help the given class to fulfil the duties.	—
GUI	Graphical User Interface — A form of user interface that allows users to interact with electronic devices through graphical icons.	interface
Input data	Data provided by the User which will provide additional information regarding processing of the input file.	—
Input file	A binary file which will be uploaded to the program in order to be processed and transformed into a readable text file.	input
Iteration	The repetition of an action	—
Output file	A result given by the program, provided that proper input data and a proper input file (or files) are given.	output
SD	Software Design — abbreviation of the document's name.	—
Software	The product developed by our Team.	program

3. CRC

The purpose of the CRC session was to establish, which classes will be used, what will be their responsibilities and what will be their collaborators, so classes which help to achieve the earlier mentioned responsibilities.
The result of this session was the following:

Data Source	
- Holds data about the data source: * path * format	

Controller	
- Validates parametes which were passed	Service *indirectly:* File Model

View	
- Generates UIs - Sends data to controller	Controller *indirectly:* Service File Model

File	
- Holds data about the file: * content * number of iterations	

Service	
- Processes the file, - Returns instance of model	File Model List of formats (?)

Model	
- Implements string interface	

Figure 1: Result of the CRC session

4. Use cases

4.1. Actors

An actor in the case of our software is any person who will have the need to convert binary data into a text file.

4.2. List of use cases

1. Choose file
2. Batch Processing (extends Choose file)
3. Specify number of iterations (extends Choose file)
4. Convert file from binary to text (includes Choose file)
5. Save text file (extends Convert file from binary to text)
6. View the result (includes Choose file and Convert file from binary to text)

4.3. Use case diagram

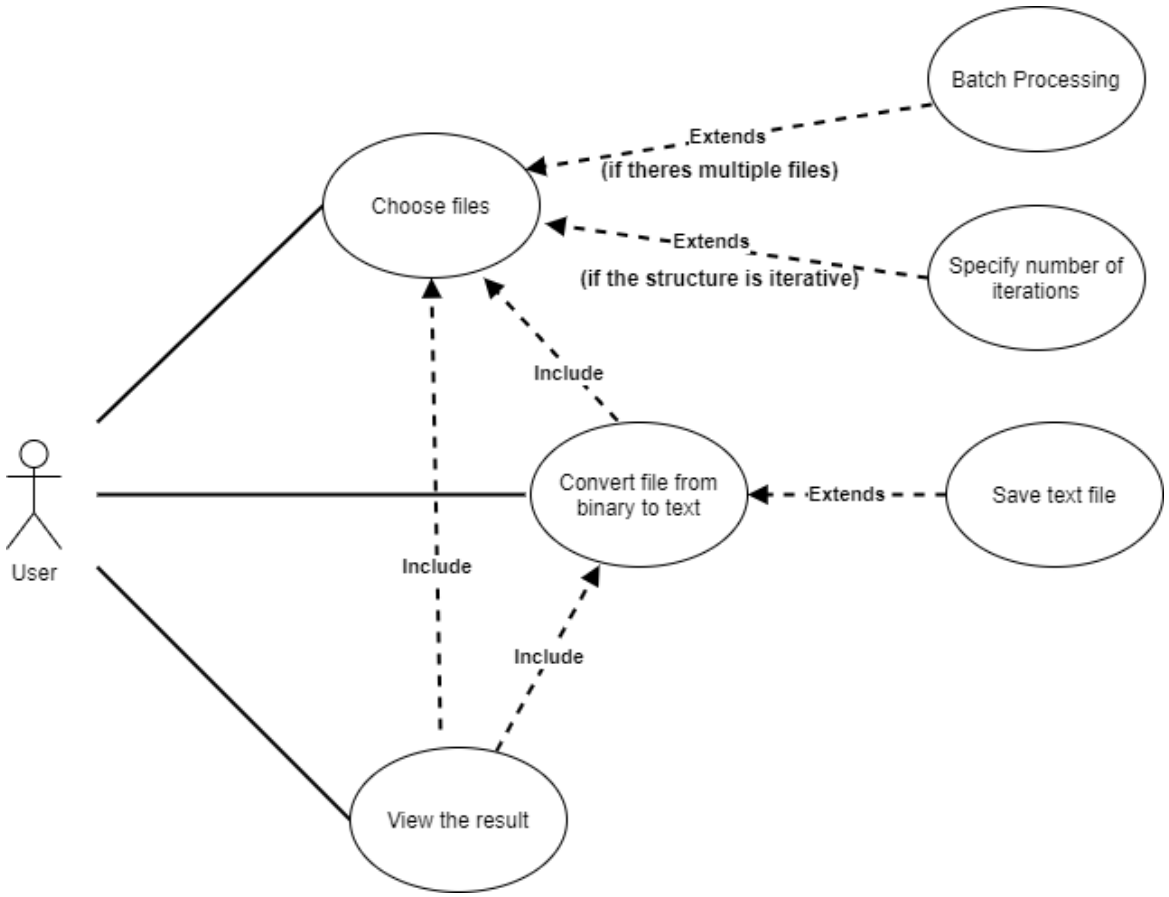

Figure 2: Use-case diagram

5. Class descriptions

Class name: RetrieveBinaryData	
Class description: Service for processing data	
Attributes	**Attribute description**
#result: BinaryInterface	Instance of class implementing Binary Interface
#path: string	Path to file
Methods	**Method description**
#get_iterative(data:bytes, iteration_number: int): Iterative	Get instance of Iterative model
#get_linear(data: bytes): Linear	Get instance of Linear model
#get_file_structure(data: bytes): string	Check if file is linear or iterative
+__init__(path: string, processing_type: string): self	Constructor
+process_data(): self	Process data to convert it from binary format to human readable format
+get_result(): bytes	Get result as binary array
+set_path(path: string)	Set path to the file

Class name: Linear	
Class description: Model implementation using linear method	
Attributes	**Attribute description**
#data: bytes	Data taken from the file
Methods	**Method description**
+get_data(): bytes: file	Get data from the file
+__init__(): self	Constructor
+to_text(): string	Convert binary data to human readable format

Class name: Iterative	
Class description: Model implementation using iterative method	
Attributes	**Attribute description**
#data: bytes	Data taken from the file
#iteration_num: int	Holds info about current iteration
Methods	**Method description**
+get_data(): bytes: file	Get data from the file
+get_iteration(): int	Get iteration number from the file
+__init__(): self	Constructor
+to_text(): string	Convert binary data to human readable format

Class name: MainController	
Class description: Validates parameters which were passed	
Methods	**Method description**
+main()	Main loop of the program
+retrieve_data(path: string, args...): string	Method calling service for processing files and returning result — file contents in human readable format
#validate(args...): bool	Check if options specified by user are valid for given file
#get_file(file_path: string): files[]	Returns files array from the given location

Class name: View	
Class description: Generates UIs, sends data to controller.	
Methods	**Method description**
+display_result(result: string)	Puts human readable result on GUI window
+render()	Updates and builds interface

6. Class diagram

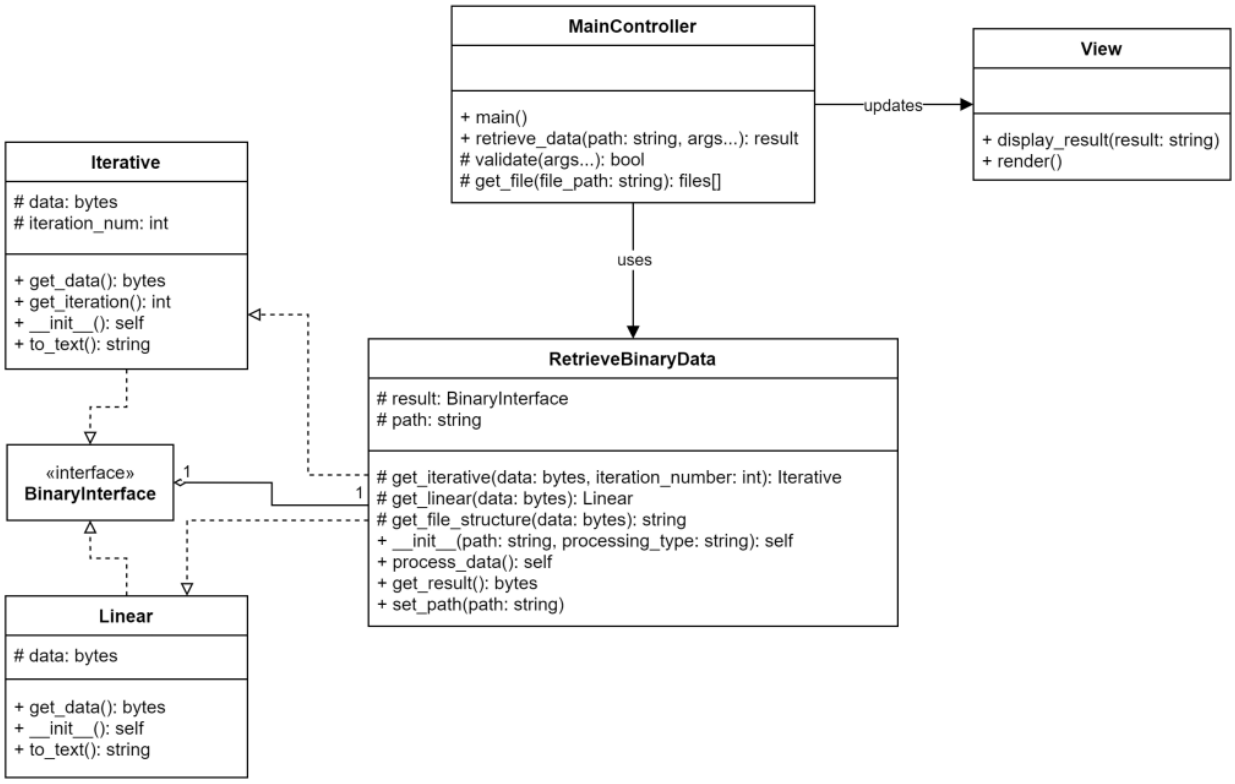

Figure 3: Class diagram

7. Sequence diagram

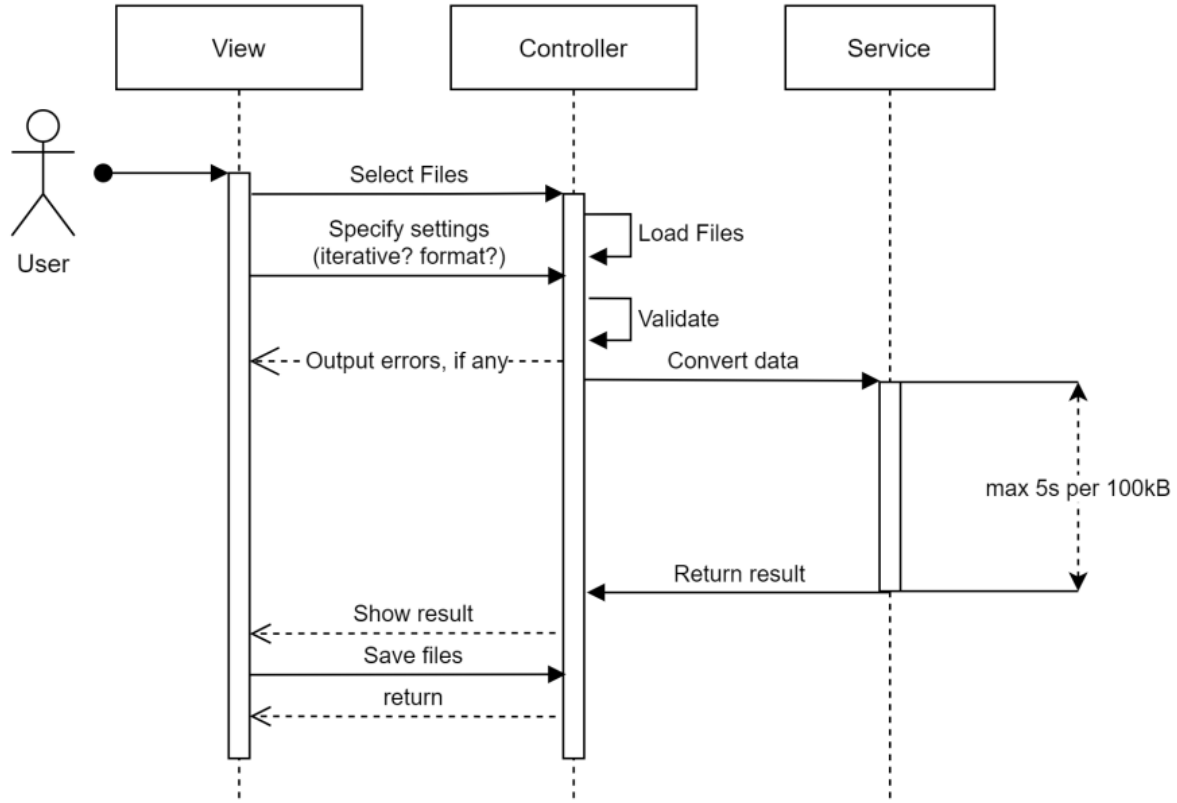

Figure 4: Sequence diagram

8. Appendices

8.1. Tools used for creating the UML diagrams

All diagrams have been created with the use of draw.io/diagrams.net.