Python Project

Requirements Analysis Document (RAD) / ITERATION 2

Purpose of the system

The purpose of the system is to develop a software for processing and analyzing Zoom poll reports. The system gets an excel file of polls, student list, and answer keys of the polls. According to answer keys, system calculates points of each student. Also, by looking at the attendance polls, system calculates the attendance of each student. The system calculates the percentages of answers given to each question and creates histograms for them. The system gives a detailed report for each student for each poll.

Stakeholders

- Murat Can Ganiz
- Lokman Altın

Scope of the system

This system can be used in online lectures to calculate attendance of students. Also it can be use in online working companies. In companies managers can increase the performance and get the employees's thoughts. The system also calculates the grade of participants in polls

Functional Requirements

• System will take a single zoom poll report as a .csv file or a folder containing several .csv files as input.

• System will take a student list.

• Program will take a single answer key or a folder containing several answer keys of different polls

• Program must detect and separate different poll results from a given file.

• Program calculate and output the attendance into an spreadsheet file (.xlsx or .ods).

• Program outputs one global file that will be maintained between different runs of our software.

• For each quiz report the "number of questions", "number of correctly answered questions", "number of wrongly answered questions", "number of empty questions", "rate of correctly answered questions" (correctly answered questions/total number of questions in the poll), "accuracy percentage" (rate of correctly answered questions*100) for each student.

• Quiz detail reports for each student: Report the details of each students answers for each student for each poll.

• The system outputs piecharts and histograms.

Non-Functional Requirements

• Histograms are shown in both output files and png format

USE CASES

• The program first reads the student list and creates an object for each of the student. These objects consist of student id , student name, student last name and emails.

• The program reads poll excel files. If input only contains attendance poll, program increases the attendance of students who are answered the poll by one.

• If there is both quiz poll and attendance poll, program increases the attendances of the students. Also, the program calculates the right answers of the students according to given answer key of each poll.

• The program calculates the points of students who answered the poll. If a student didn't answer the poll questions or didn't enter the lecture, her/his point is assigned to 0.

• According to answers of the questions, histograms are created.

• The attendances of students are rated to the total number of lectures.

• In each reading of a quiz poll file, the results are combined in one global file.

- The system creates excel files for each student. The files contain questions, given answers and correct answers and 0-1 according the answers. The system collects these files into a folder and updates them in each run.
- The program continues to work like that.

Glossary of Terms

Poll - Questions that are ask by the teacher in lectures in any time

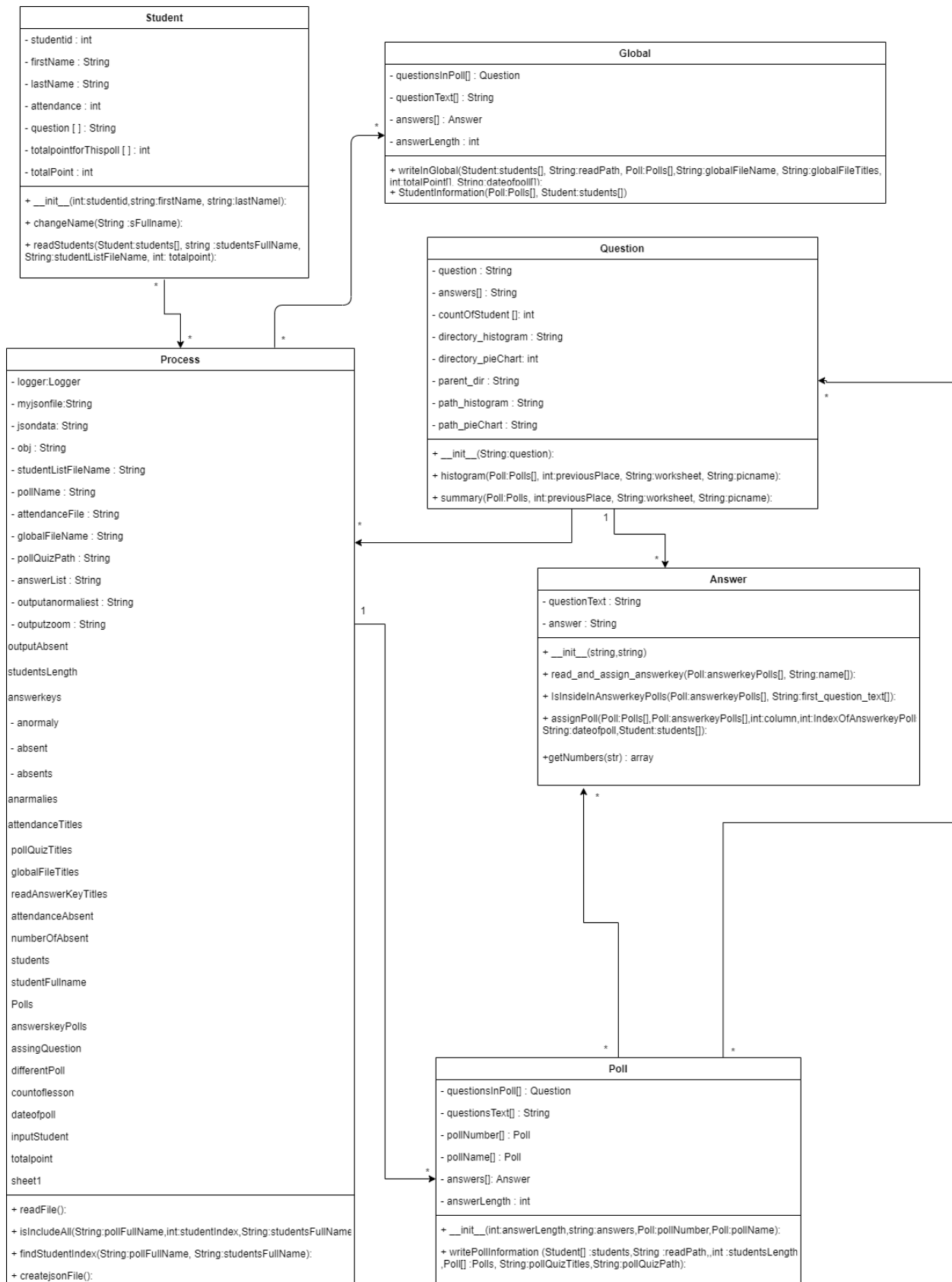*Attendance Poll –* Questions that are asked to take the attendance

Quiz Poll – Questions that are asked about class subjects

Student – People who are answering the polls

Histogram – The diagram that shows the percentages of answer given in each poll questions

Global File - The file containing all of the points of the students

Domain Model :

Student

- studentid : int
- firstName : String
- lastName : String
- attendance : int
- question [ ] : String
- totalpointforThispoll [ ] : int
- totalPoint : int

+ __init__(int:studentid,string:firstName, string:lastNameI):
+ changeName(String :sFullname):
+ readStudents(Student:students[], string :studentsFullName, String:studentListFileName, int: totalpoint):

Global

- questionsInPoll[] : Question
- questionText[] : String
- answers[] : Answer
- answerLength : int

+ writeInGlobal(Student:students[], String:readPath, Poll:Polls[],String:globalFileName, String:globalFileTitles, int:totalPointI, String:dateofpollI]):
+ StudentInformation(Poll:Polls[], Student:students[])

Process

- logger:Logger
- myjsonfile:String
- jsondata: String
- obj : String
- studentListFileName : String
- pollName : String
- attendanceFile : String
- globalFileName : String
- pollQuizPath : String
- answerList : String
- outputanormaliest : String
- outputzoom : String
outputAbsent
studentsLength
answerkeys
- anormaly
- absent
- absents
anarmalies
attendanceTitles
pollQuizTitles
globalFileTitles
readAnswerKeyTitles
attendanceAbsent
numberOfAbsent
students
studentFullname
Polls
answerskeyPolls
assingQuestion
differentPoll
countoflesson
dateofpoll
inputStudent
totalpoint
sheet1

+ readFile():
+ isIncludeAll(String:pollFullName,int:studentIndex,String:studentsFullName
+ findStudentIndex(String:pollFullName, String:studentsFullName):
+ createjsonFile():

Question

- question : String
- answers[] : String
- countOfStudent [] : int
- directory_histogram : String
- directory_pieChart: int
- parent_dir : String
- path_histogram : String
- path_pieChart : String

+ __init__(String:question):
+ histogram(Poll:Polls[], int:previousPlace, String:worksheet, String:picname):
+ summary(Poll:Polls, int:previousPlace, String:worksheet, String:picname):

Answer

- questionText : String
- answer : String

+ __init__(string,string)
+ read_and_assign_answerkey(Poll:answerkeyPolls[], String:name[]):
+ IsInsideInAnswerkeyPolls(Poll:answerkeyPolls[], String:first_question_text[]):
+ assignPoll(Poll:Polls[],Poll:answerkeyPolls[],int:column,int:IndexOfAnswerkeyPoll String:dateofpoll,Student:students[]):
+getNumbers(str) : array

Poll

- questionsInPoll[] : Question
- questionsText[] : String
- pollNumber[] : Poll
- pollName[] : Poll
- answers[]: Answer
- answerLength : int

+ __init__(int:answerLength,string:answers,Poll:pollNumber,Poll:pollName):
+ writePollInformation (Student[] :students,String :readPath,,int :studentsLength ,Poll[] :Polls, String:pollQuizTitles,String:pollQuizPath):

CREATED BY GROUP:

- 150117034 Berra Mercan

- 150117510 Elif Gökpınar
- 150117042 Ezgi Doğruer
- 150117061 İsra Nur Alperen
- 150117057 Rümeysa Öztürk
- 150116010 Şükriye Soyer
- 150114036 Ömer Kaya