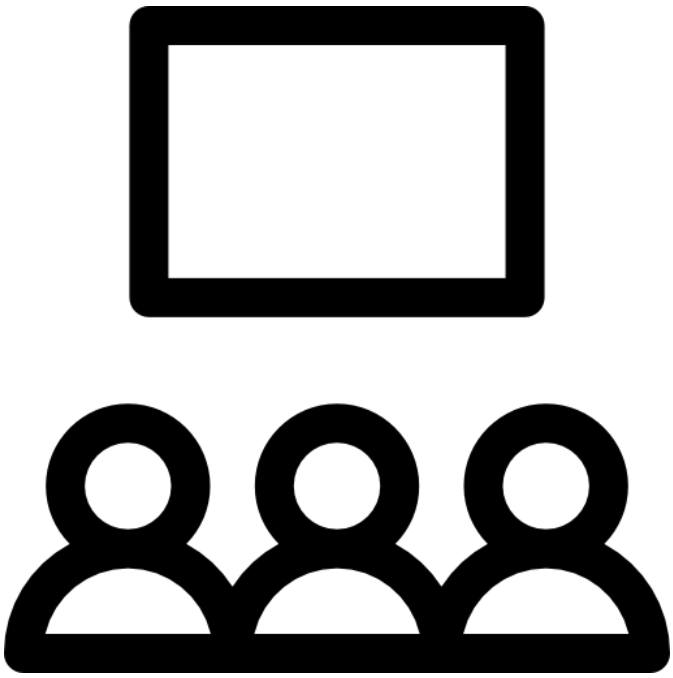
IT PAT
Specs and Design Document

Classroom Manager



Name: Clifton Bartholomew

Grade: 24

Contents Page

1.1. Problem summary

½ - 1 page

1.2. Motivation and Research

1.2.1. Existing solution

1.2.2. How my project will differ from the current available programs

+- 1 page

1.3. Specs of program functionality

1.3.1. General specifications

- Each screen should be able to navigate back to the home screen.
- Each section needs its own permanent storage file (and a corresponding backend manager).

1.3.2. Main home screen section

- The user must be presented with a navigation menu of buttons/menu's to direct them to the appropriate screen.

1.3.3. Manage students section

- The user should be presented with a list of the current students.
- The user should be able to type in a student name and surname and be able to delete that student or add that student to the list in permanent storage.
- A student needs a name and a surname.

1.3.4. Manage assessments section

- The user should be presented with a list of the current assessments.
- The user should be able to type in an assessment name and type and be able to delete or add that assessment to the list in permanent storage.

1.3.5. Manage records section

- The user should be presented with a list of the current students.
- The user should be able to select from a list of the current assessments.
- The user should be able to enter a mark for the assessment.
- The record of the student mark should be updated in permanent storage.

1.3.6. View records section

- The user should be able to select a student and view all the records for that student.
- The user should also see the students average for all the assessments.

1.4. Specs of data storage

1.4.1. Students

- **Fields:** name and surname
- **When are records created:** in the “add new student” part of the student management section
- **When are records accessed:** when bringing up a list of current students for display in the manage student and manage records sections
- **When are records updated:** in the “add/delete new student” part of the student management section.

1.4.2. Assessments

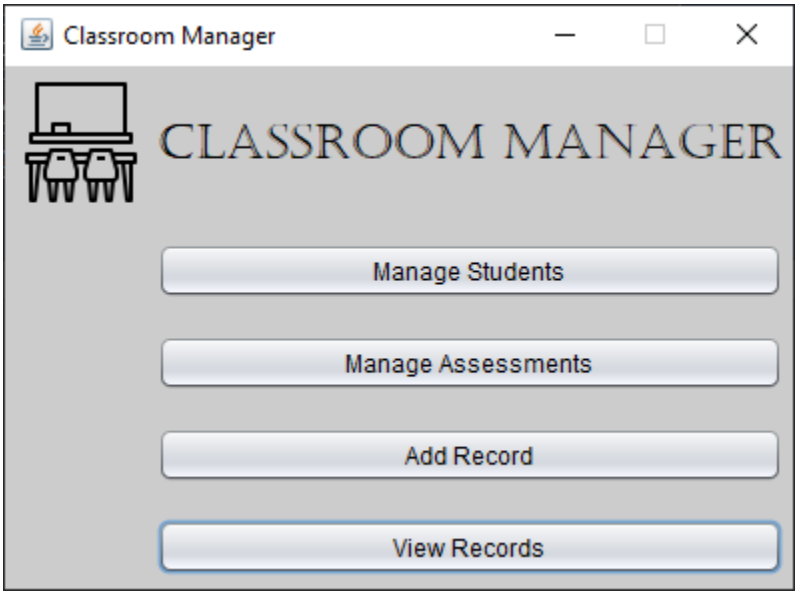
- **Fields:** assessment name and type
- **When are records created:** in the “add new assessment” part of the student management section
- **When are records accessed:** when bringing up a list of current assessments for display in the manage assessment and manage records sections.
- **When are records updated:** in the “add/delete new assessment” part of the assessment management section.

1.4.3. Records

- **Fields:** student, assessment and percentage
- **When are records created:** in the “add new record” part of the add record section
- **When are records accessed:** when bringing up a list of records for a specific student in the add record section.
- **When are records updated:** in the “add new record” part of the add record section.

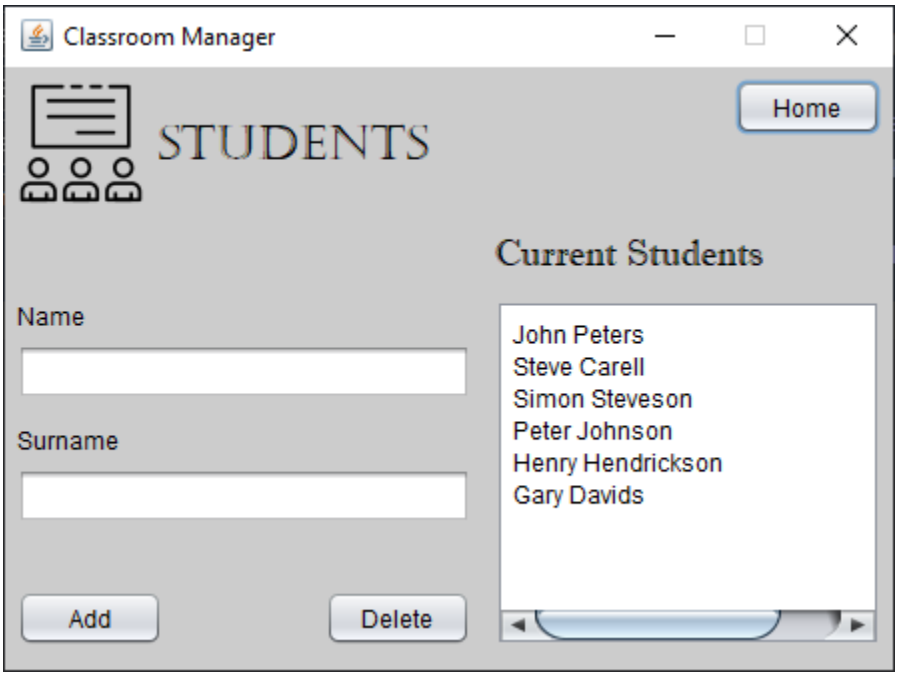
2.1. Interface design

2.1.1. Main home screen - MainMenuUI



Description	The purpose of this screen is for navigation between all the other screens.
Data In	• An image is retrieved from the resources folder to be displayed next to the heading.
Actions	**Manage students button** • The user clicks this and is redirected to the Students section. **Manage assessment button** • The user clicks this and is redirected to the Assessments section. **Add record button** • The user clicks this and is redirected to the Add Records section. **View records button** • The user clicks this and is redirected to the View Records section.

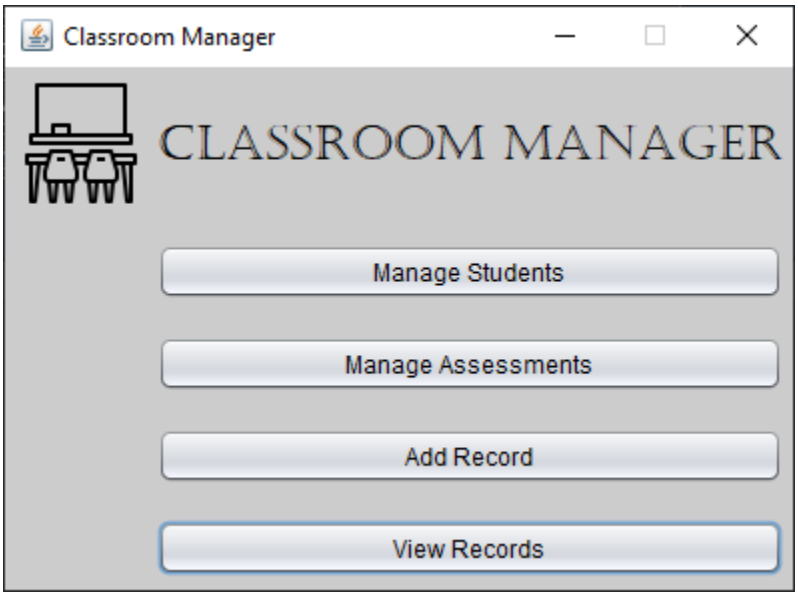
2.1.2. Manage students section - ManageStudentsUI



Description	The purpose of this screen is to manage the list of students stored in permanent storage.
Data In	• An image is retrieved from the resources folder to be displayed next to the heading. • The list of students will be retrieved through the **StudentManager** class which accesses the students from a **students.txt** file.
Data Out	• The name and surname entered into the text fields are passed to the **StudentManager** which updates the text file by either deleting or adding that student.
Actions	**Add button** • If there is a name and surname, sends the name and surname written in the text fields to the StudentManager class to be added to the students.txt file **Delete button** • If there is a name and surname, sends the name and surname written in the text fields to the StudentManager class to be removed from the students.txt file **Home button** • Returns to the main menu.

2.2. Program flow

2.1.1. Home Screen - MainMenuUI



- When **Manage Students** button is pressed

```
MainMenuUI is disposed()

ManageStudentsUI is launched
```

- When **Manage Assessments** button is pressed

```
MainMenuUI is disposed()

ManageStudentsUI is launched
```

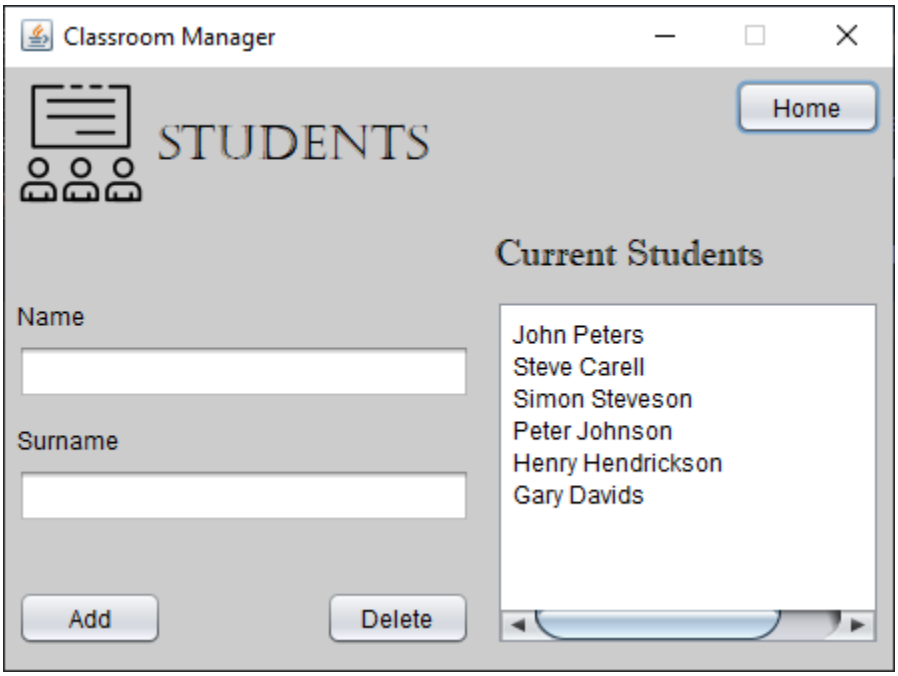
- When **Add Record** button is pressed

```
MainMenuUI is disposed()

ManageStudentsUI is launched
```

- When **View Records** button is pressed

```
MainMenuUI is disposed()

ManageStudentsUI is launched
```

2.1.2. Students manager screen - ManageStudentsUI



- When the screen is **initialised**

```
Make a call to the StudentsManager class.
Scanner sc <- students.txt
while sc.hasNext()
     output <- sc.nextLine()
display output in the textArea
```

- When the **Add** button is pressed

```
fullName is retrieved from UI textFields.
fullname is passed to StudentsManager class.
Printwriter pw <- students.txt (with append)
pw.println(fullname)

Make another call to the StudentsManager class.
Scanner sc <- students.txt
while sc.hasNext()
     Output <- sc.nextLine()
display output in the textArea
```

- When the **Delete** button is pressed

```
fullName is retrieved from UI textFields.
fullname is passed to StudentsManager class.
Scanner sc <- students.txt
while sc.hasNext()
     currentStudent <- sc.next()
     if fullname != currentStudent
          output <- currentStudent
Printwriter pw <- students.txt (with NO append)
pw.print(output)

Make another call to the StudentsManager class.
Scanner sc <- students.txt
while sc.hasNext()
     output <- sc.nextLine()
display output in the textArea
```

- When the **Home** button is pressed

```
The MainMenuUI is launched
ManageStudentsUI is disposed
```

2.3. Class Design

AssessmentManager	Description
-fileName : string	The path to assessments.txt
+getAssessments() : string +getAssessmentsAsArray() : string [] -getNumAssessments() : int +addAssessment(assessmentName: string, assessmentType: string) +deleteAssessment(assessmentName: string)	Gets all assessments in a multilined string Gets all assessments in an array of strings Helper method to get the number of assess Save an assessment to assessments.txt Delete an assessment from assess.txt

StudentManager	Description
-fileName : string	The path to students.txt
+getStudents() : string +getStudentsAsArray() : string [] -getNumStudents() : int +addStudent(name: string, surname: string) +deleteStudent(name: string, surname: string)*(to add, must call deleteRecord(name, surname)	Gets all students in a multilined string Gets all students in an array of strings Helper method to get the number of studen Save an assessment to students.txt Delete an assessment from students.txt

RecordManager	Description
-fileName : string	The path to records.txt
+getStudentRecords(student : string) : string +addRecord(name : string, assessment : string, percentage : string) +deleteRecord(name : string, assessment : string) +deleteRecords(name : string)	Gets all records for a student in a multilined string Saves a record to the records.txt file Deletes a record in the records.txt file Deletes all records for a specific student

2.4. Secondary storage design

.2.4.1. students.txt

Format per line

<Name> <Surname>

Example

John Peters Steve Carell Simon Steveson Peter Johnson Henry Hendrickson Gary Davids

.2.4.2. assessments.txt

Format per line

<assessment name>#<assessment type>

Example

Term 1 Theory Test#Theory Term 2 Practical Test#Practical Term 3 Practical Test#Theory Term 2 Theory Test#Theory Term 2 June Exam P1#Theory

.2.4.3. records.txt

Format per line

<name> <surname>#<assessment name>#<mark>

Example

Henry Hendrickson#Term 1 Theory Test#85 Simon Steveson#Term 1 Theory Test#50 Henry Hendrickson#Term 2 Theory Test#55 Simon Steveson#Term 3 Practical Test#66