Criterion A: Planning

Defining the problem

The client Mr. xx works as a dorm parent in an international boarding school. Every dinner, he has to carry a paper sheet to sign in every student to keep in track that they are all there. With other prefects, we notice that it is very inconvenient and a waste of paper to sign in students with a sheet every day. Moreover, it will be a waste of space to keep piles of sign in sheet to see the record of how many times the students have been late for at the end of the school year. With a chance to solve real problem with my knowledge in computer science, I decided to create a program with a user interface that can let students sign in with computer and keep the late data digitally.

I went to Mr. xx to be aware of what kind of features are needed for this sign-in program. Xx told me that he wants to use key fob as a media for student to sign in; however, as he understands that requires a program that connects to other peripheral devices, xx told me to let student input their student ID number to as the input to sign in.

I asked my computer science teacher about this idea and he agreed. Therefore, I decided to help xx by creating a sign-in software.

After talking to xx about what he is looking forward to this software, we agreed to create a program with three sections: for students to sign in, for dorm parents to access student's accumulated late times, and for dorm parents to change student's information. The list of students who are absent from the dinner after the end of sign-in section will be returned for dorm parents to see who is late.

Following my computer science teacher's advice, I created a flow chart before actual coding to have a clearer picture of my software.

Rationale for the proposed solution

Java is the computer language that my school teaches, so I decided to use Java to write the program. Java allows me to make a user interface that will satisfy my client without over challenging my ability in coding. Moreover, java can be run on every platform which is an advantage if other dorm parents use platforms other than OS. Xx does not have any basic computer science skill, so it is better to make the program easy to manage and use. To give the most comfortable interface, Java gives the best solution by having the ability to create controls with ActionListener, JLabel, JButton, JTextField, and so on. As the program needs to be initiated with every student's information first, I need to create a Student class to hold the information. It is also possible that with Scanner, I can then scan through a student.txt to

input the parameters to create a Student object and store each Student object in a HashMap. Moreover, the student list and accumulated late time should be saved in file.

Stating success criteria

- The program should have three buttons/options on the home window: 1st -for students to sign in; 2nd – for dorm parents to get accumulated late times; 3rd – for dorm parents to change student information
- The program allows students to sign in by inputting student ID number
- If an invalid number is inputted, info message will appear on the screen
- The program allows xx to see a new window with a list of students who are absent when end the sign-in program
- The program allows xx to modify student's information
- Xx can share to program with other dorm parents who are on duty that day
- There will be s student data base created by teacher to input as the information hold in Student class before the program starts
- The program allows the student's accumulation late time digitally and easy to access and share among dorm parents
- In case of data entry errors, info messages will appear on the screen.

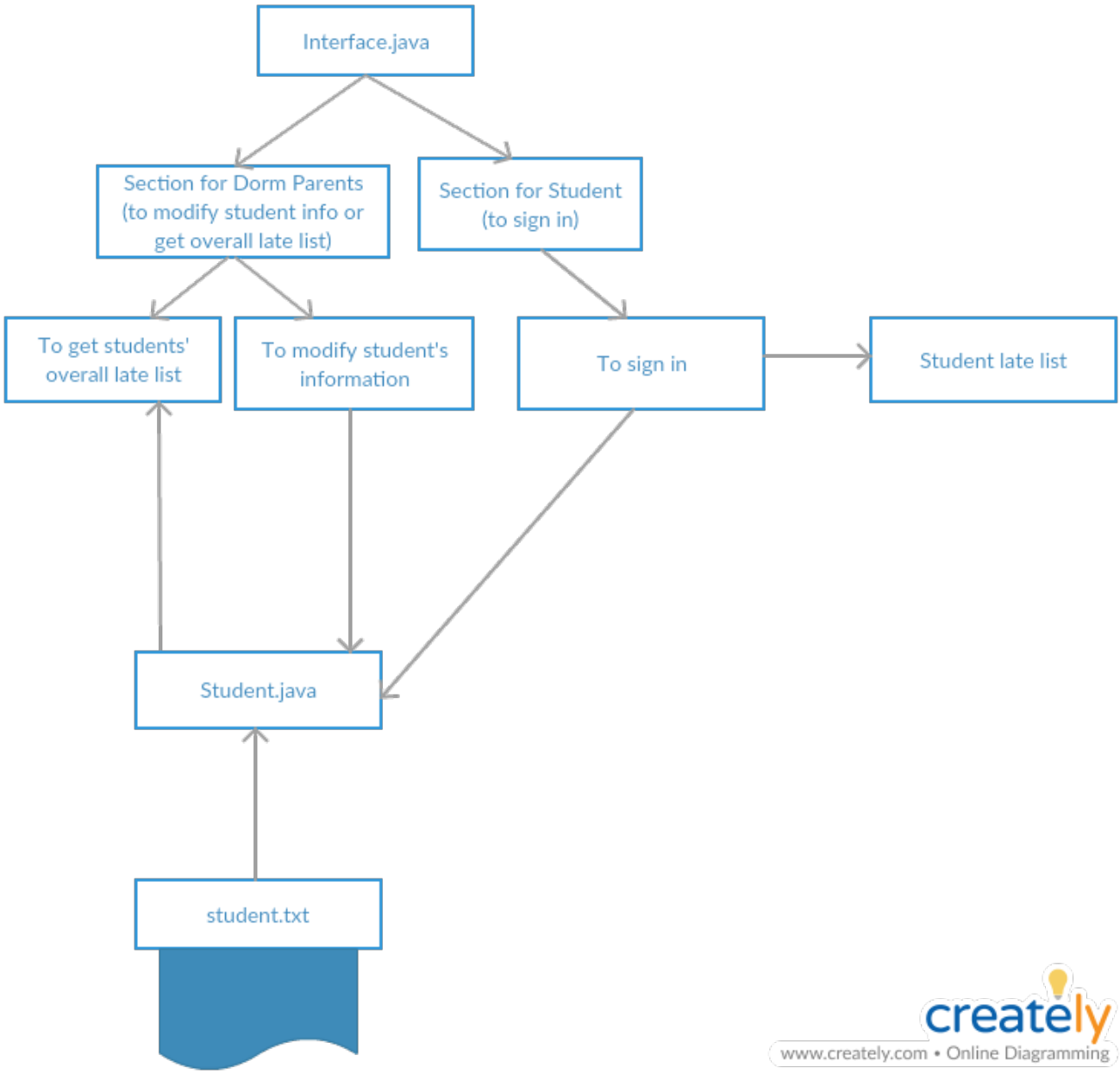
Criterion B: Record of tasks

Task number	Planned action	Planned outcome	Time estimated	Target completion date	Criterion
1	Choosing a topic and discuss idea with future client	Decided the problem and confirm with the willingness of Ms. Xx to collaborate as the client of my project	2 days		A
2	Discuss idea with computer science teacher	Teacher approved the idea	3 days		A
3	First interview with Mr. xx	Clear description of the problem	2 days		A
4	Confirm my solution for the client's problem with computer science teacher	Java chosen to be used as a solution for client	2 days		A
5	Define criteria for success and finish with Criterion A	Criteria for success is defined and finished Criterion A	2 days		A
6	Thinking on design of a product	Start developing basic design and create a flow chart	1 week		B
7	Create a schedule for developing the software	Develop schedule in discussion with Mr. xx about the software	1 week		B
8	Work on interface	Finish designing interface and agreed with Mr. xx	1 week		B/C
9	Draw diagrams and flow charts to clarify the program working process	Finish diagrams and process description flow charts; finish Criterion B	1 week		B
10	Think about a test to check whether the program works	Finished Detailed test plan	1 week		B

11	Start writing actual code	Finished source code of Student class and SignIn class	1 week		C
12	Finding more information about ActionListner and Scanner methods	Researched information from the internet	5 days		C
13	Develop code and test every time to assure that everything works fine	Different classes and methods are added in whole program with that makes the code more readable; function testing should be made in whenever a change is made	4 weeks		C
14	Finish writing criterion C	Finished extended writing together with the source code	1 week		D
15	Test the final product solution	Show Mr. xx the final product	2 days		D
16	Interview with Ms. xx	Get fedback from Ms. xx: testing results and suggestions about further developments	1 day		A/ E
17	Think about how to further improve the program	Written ideas on how to improve program in future	1 day		A/D/E
18	Record video to show programs functionality	Video is recorded successfully	1 day		D

Criterion B: Design

Program basic (initial) structure:



Input data (Dorm parents)

- Dorm parents can either choose to modify student information or get students' overall late list
- When choose to modify student information, dorm parents have to first input the student's ID number

Ex: Normal data	Ex: Abnormal data Student ID number: 13ueydghbkasdkv

Student ID number: 1111111 (must be seven-digit)	

Limitation: Dorm parents can only modify one student information at a time. Also, dorm parents cannot separate overall late list of a student, but every student's overall late list.

- Student.txt must be typed in this specific way

Ex: Normal data	Ex: Abnormal data
Red(tie color) 1111111(ID number) Nerissa(name) 0(total late time)	1111111 Red 1 Nerissa

Input data (Students action)

- Data- students will input their ID number in order to sign in

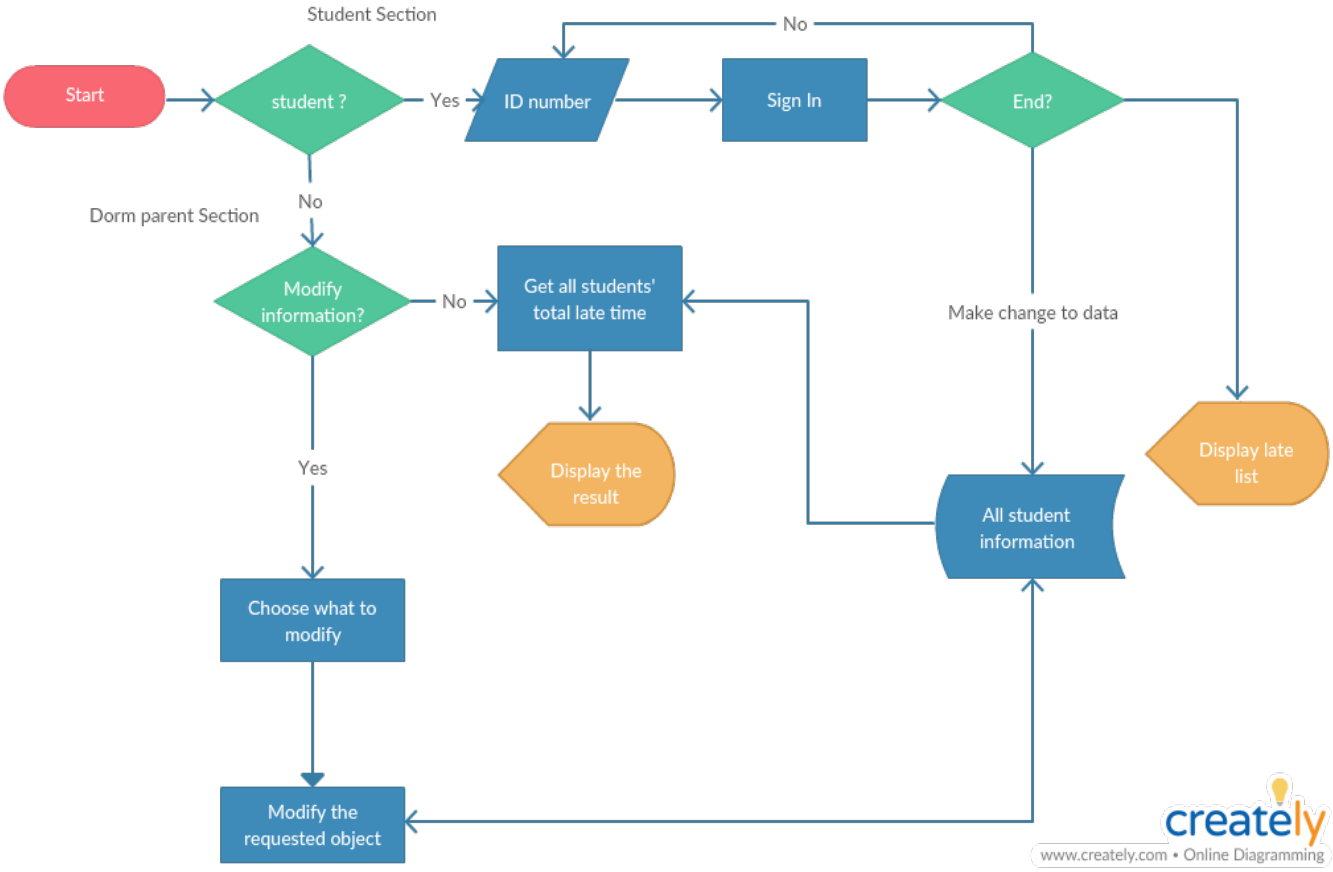
Ex: Normal data	Ex: Abnormal data
Your student ID number: 1111111	Your student ID number: 21ssdkvosdljv

When student input their ID number, their late status variable in their Student object will change to false, so that later they will not appear in the returning late list.

Output data

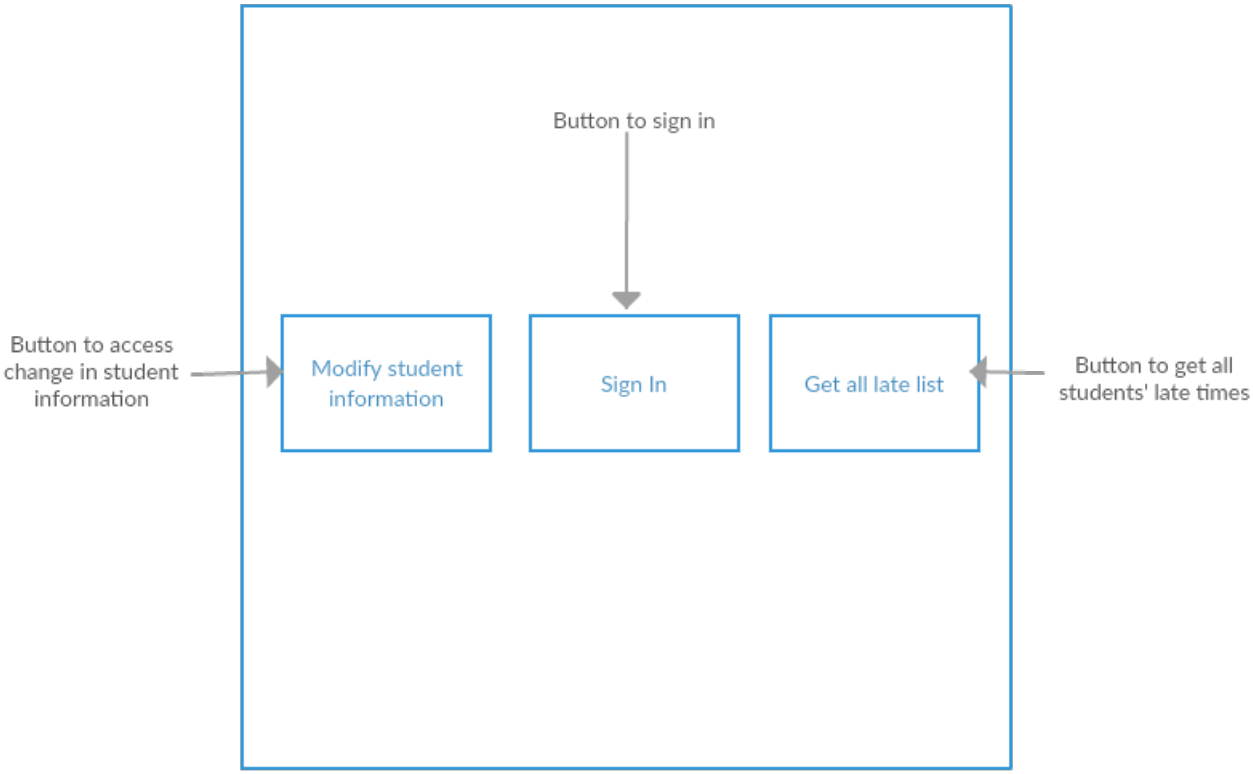
Dorm parent's section	Student Section
1. a student late list in JTextArea will appear in a new window (can be printed) 2. student information can be modified as requested	1. When the student sign in, they change their status in their information to false, so late they will not later identify as late

Flow Chart


Student Section
Start
student ?
Yes
ID number
Sign In
End?
No
Dorm parent Section
No
Modify information?
No
Get all students' total late time
Make change to data
Display late list
Display the result
Yes
All student information
Choose what to modify
Modify the requested object
creately
www.creately.com • Online Diagramming

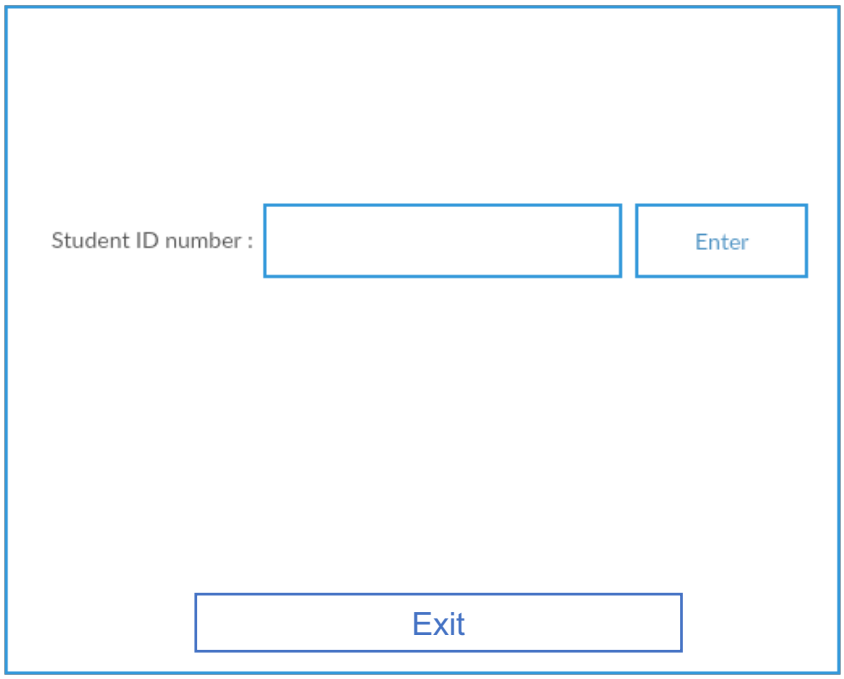
Software Interface

- Home window

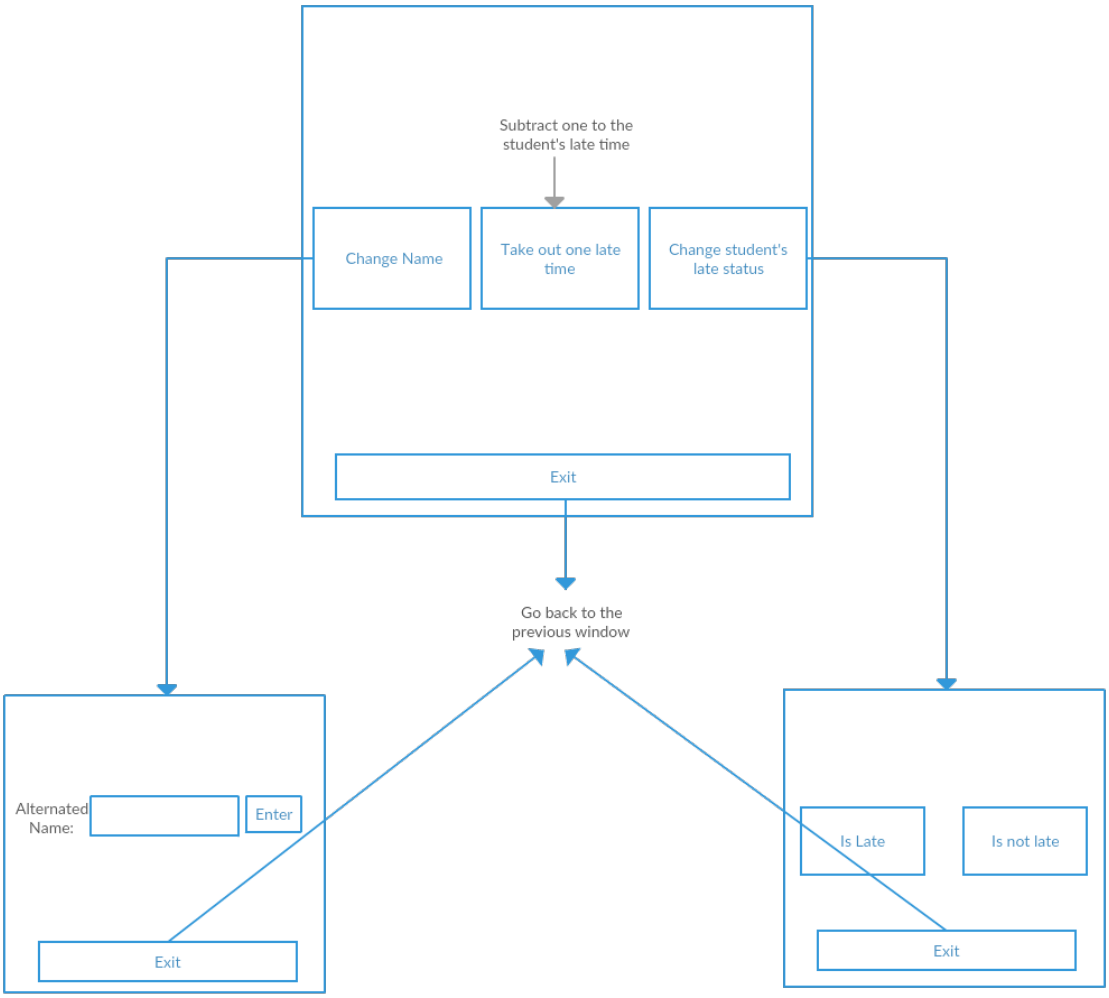


- Modify student information (for dorm parent's section)

Dorm parents have to first clarify the student whose information they want to make change of.

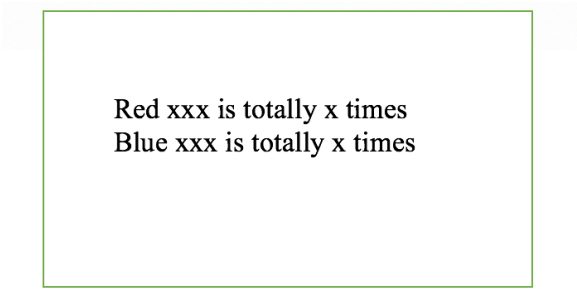


Then, the dorm parents can pick what they want to change among three options (requested by the client)

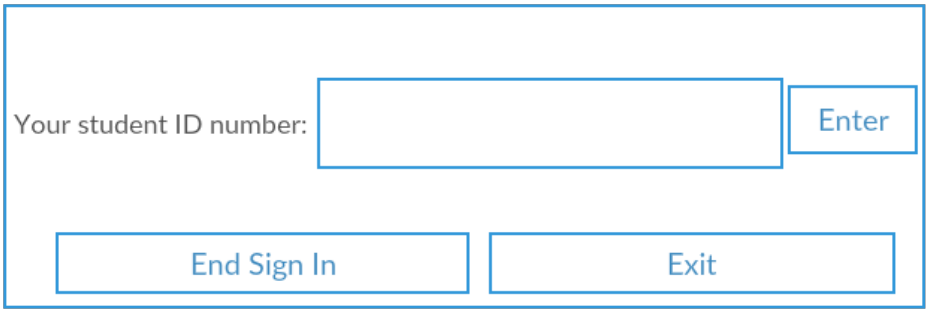


Get all late list (for dorm parent's section)

- Result:



Sign in (for student's section)



Schedule for developing the program

Program will be divided in three sections; Modify Student Information section for dorm parents to make change in student's information; Get All Late List section for dorm parents to change the student's late status; Sign In section for students to sign in for dinner

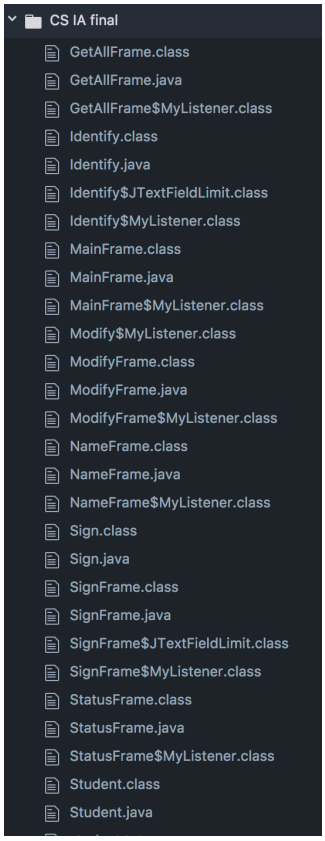
2 + 3 weeks will be enough to complete all sections:

Modify Student Information (for dorm parent's section)	Get All Late List (for dorm parent's section)	Sign In (for student's section)
• Create interface • Write a method that can find the student file by inputting the student's ID number • Write code for exit button that allows user to go back to the previous window • Write code to prevent dorm parents from data entry errors	• Create interface • Write code that gets every student late time from the student file stored in Student object • Write code for exit button that allows user to go back to the previous window • Write a code that create a new window to display the data	• Create interface • Write code that will hold every student's information from student.txt using Scanner and Student class • Write code for changing student late status and record the students who are late into an ArrayList • Write code for end sign button that create a new window that displays student who are absent list • Write code for exit button that allows user to go back to the previous window

Action Test	Action Test	Action Test
Test if the program runs correctly and appears on the main window	Test if the program runs correctly and appears on the main window	Test if the program runs correctly and appears on the main window
Check if the button works and leads to the right window	Check if the button works and leads to the right window	Check if the button works and leads to the right window
Check if the requested change is done and changed from the student.txt	Check if all the data is correct and compare it with the data in student.txt	Check if the students who sign in do not appear on the late list and appear those who supposed to do
Check if the program catches invalid student ID number	Check if the program has a proper size	Check if the program catches invalid student ID number

Criterion C: Development

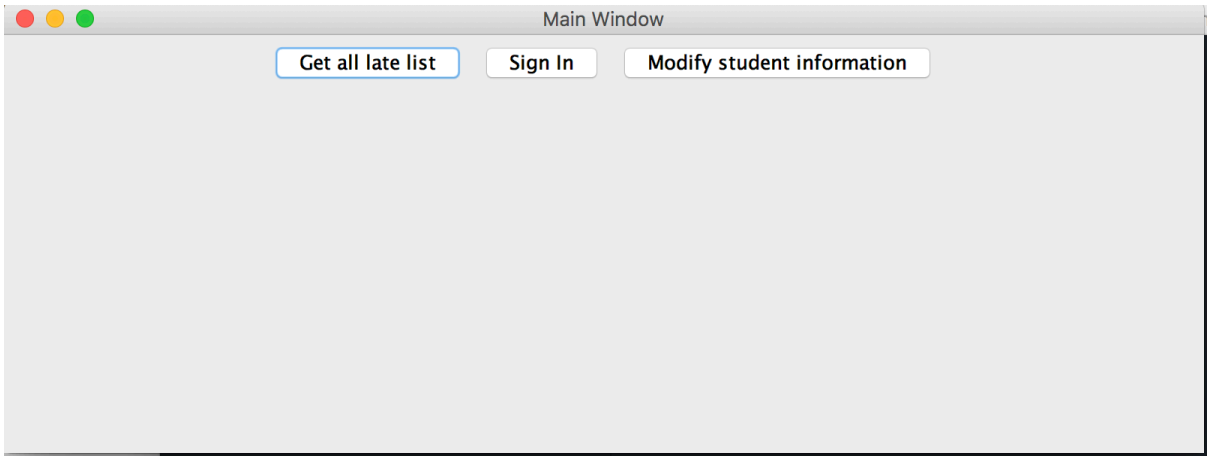
All classes:



List of the techniques:

1. Each class extends JFrame to create controls and place them in the window
2. I created my own JFrame class and made my own methods in different frame classes for their different purposes
3. for loop
4. HashMap and ArrayList
5. JComponent and Panels
6. ActionListener
7. JTextFieldLimit
8. Scanner and File: student.txt
9. Static methods and variables

Main Window:



MainFrame class extends JFrame:

Use of instance variable:

```
private static JButton getLate_btn= new JButton("Get all late list");
private static JButton sign_btn= new JButton("Sign In");
private static JButton modify_btn= new JButton("Modify student information");
```

Use of constructor:

```
public MainFrame(){
  createInteractors();
  setTitle("Main Window");
  setSize(800,300);
  setDefaultCloseOperation(EXIT_ON_CLOSE);
  setVisible(true);
}
```

createInteractors method:

```
public void createInteractors(){
  MyListener listener= new MyListener();
  //layout
  Panel pnl=new Panel();
  add(pnl, BorderLayout.CENTER);

  //components
  pnl.add(getLate_btn);
  getLate_btn.addActionListener(listener);

  pnl.add(sign_btn);
  sign_btn.addActionListener(listener);

  pnl.add(modify_btn);
  modify_btn.addActionListener(listener);

}
```

MyListener class implements ActionListener:

- Add meaning and instructions to the buttons so that based on which button the user is able to decide what to do.

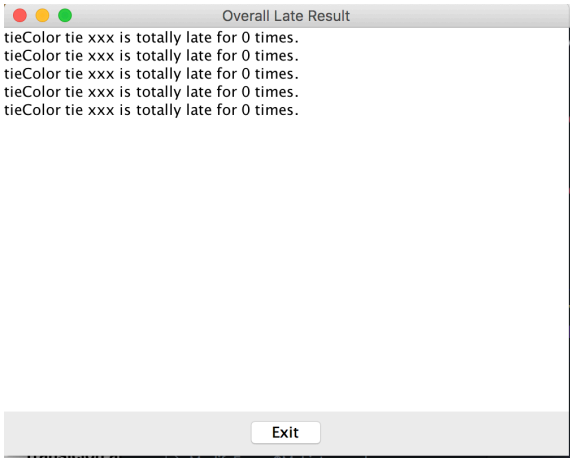
```
public class MyListener implements ActionListener{
  @Override
  public void actionPerformed(ActionEvent e){
    if(e.getSource()==getLate_btn){
      dispose();
      GetAllFrame getAllFrame=new GetAllFrame(sign);
      getAllFrame.setVisible(true);
      //getAllFrame.createInteractors();

    }
    else if(e.getSource()==sign_btn){
      dispose();
      SignFrame signFrame=new SignFrame(sign);
      signFrame.setVisible(true);
      //signFrame.createInteractors();
    }
    else if(e.getSource()==modify_btn){
      dispose();
      Identify identify =new Identify(sign);
      identify.setVisible(true);
      //ModifyFrame modifyFrame=new ModifyFrame();
    }
  }

}
```

actionPerformed method:

- Clarifies what action to be taken for each button

Overall Late Result Window:

When clicked “Get all late list”, a GetLateFrame object variable is declared and create the following window with an “Exit” button:



GetLateFrame class extends JFrame:

- The window shows the total times a student has been late for dinner in a period of time. The “Exit” button will allow users to go back to the Main Window.

```
public class GetAllFrame extends JFrame{
    private static JButton exit_btn=new JButton("Exit");
    private Sign sign;

    public GetAllFrame(Sign sign){
      this.sign=sign;
      createInteractors();
      setTitle("Overall Late Result");
      setSize(500,400);
      setDefaultCloseOperation(EXIT_ON_CLOSE);
      setVisible(true);
    }

    public void createInteractors(){
      MyListener listener=new MyListener();
      String text= sign.getAll();
      JTextArea area= new JTextArea(text);
      area.setEditable(false);
      add(area,BorderLayout.CENTER);
      Panel pnl=new Panel();
      pnl.add(exit_btn);
      add(pnl, BorderLayout.SOUTH);
      exit_btn.addActionListener(listener);
    }
```

MyListener class implements ActionListener:

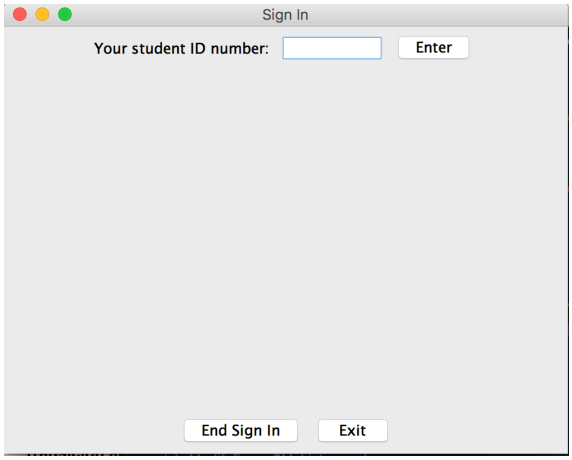
actionPerformed method:

- Allow users to go back to the Main Window

```
    public class MyListener implements ActionListener{

      @Override
      public void actionPerformed(ActionEvent e){
        if(e.getSource()==exit_btn){
          dispose();
          MainFrame mainFrame=new MainFrame();
        }
      }
    }
```

Sign In Window:

When click “Sign In,” a SignFrame object variable is declared to display the following interface window with a text field that the user can type numbers and three buttons: “Enter,” “End Sign In,” “Exit.”



SignFrame class extends JFrame:

The use of private variables for the JComponents in the interface:

```
private static JButton enter_btn=new JButton("Enter");
private static JButton end_btn=new JButton("End Sign In");
private static JButton exit_btn=new JButton("Exit");
private static JTextField sid_text= new JTextField(7);
private static JLabel sid_label=new JLabel("Your student ID number: ");
private Sign sign;
```

The use of constructor to set the interface size and to be visible:

```
public SignFrame(Sign sign){
  this.sign=sign;
  createInteractors();
  setTitle("Sign In");
  setSize(500,400);
  setDefaultCloseOperation(EXIT_ON_CLOSE);
}
```

createInteractors method:

- Set up the JComponents in the window and add ActionListener to add meaning and instruction to each button

```
public void createInteractors(){
  MyListener listen= new MyListener();

  Panel pnl1=new Panel();
  Panel pnl2=new Panel();
  add(pnl1, BorderLayout.CENTER);
  add(pnl2, BorderLayout.SOUTH);

  pnl1.add(sid_label);

  sid_text.setDocument(new JTextFieldLimit(7));
  sid_text.setActionCommand("Enter");
  sid_text.addActionListener(listen);
  pnl1.add(sid_text);

  enter_btn.addActionListener(listen);
  pnl1.add(enter_btn);
```

```
    end_btn.addActionListener(listen);
    pnl2.add(end_btn);

    exit_btn.addActionListener(listen);
    pnl2.add(exit_btn);

  }
```

MyListener class implements ActionListener:

- implements from ActionListener add instructors for each JButton when click.

```
public class MyListener implements ActionListener{
  private JFrame frame;

  @Override
  public void actionPerformed(ActionEvent e){
    //when click enter or pressed "return"
    if(e.getSource()==enter_btn || e.getSource()==sid_text){
      //get the input ID nnumber in the text field
      int id=Integer.parseInt(sid_text.getText());
      //sign in
      sign.signIn(id);
      //clear out the text field for new input
      sid_text.setText("");
    }
    //when click end
    else if(e.getSource()==end_btn){
      //stop students from signing in & create late list
      sign.endSign();
      setVisible(false);
      frame=new JFrame("These people are not here");
      JTextArea area= new JTextArea(sign.getLateList().toString());
      area.setEditable(false);
      frame.add(area);
      frame.setSize(300,400);
      frame.setDefaultCloseOperation(frame.EXIT_ON_CLOSE);
      frame.setVisible(true);
      frame.add(exit_btn, BorderLayout.SOUTH);
      exit_btn.addActionListener(this);
      //clear out the list for the next sign in
      sign.clearLateList();
    }
```

Sign class:

- I create a Sign class to handle student sign in when the user input a student number in the JTextField I have declared.
- The constructor catches when the text file does not exit by printing out “Do not find file.”

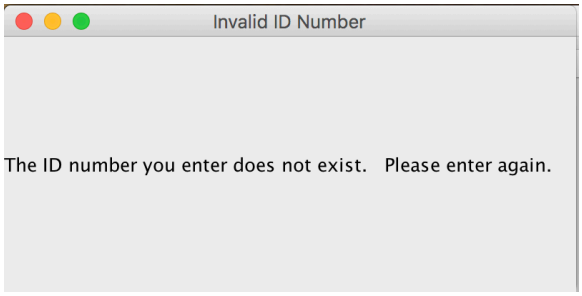
```java
public class Sign{
  //holds the list of students who are late
  private ArrayList<String> lateList;
  //holds each student's file and can be searched by ID number
  private HashMap<Integer, Student> stuList;

  //Class Sign constructor
  public Sign(){
    try{
      stuList=new HashMap<>();
      lateList=new ArrayList<>();
      Scanner input= new Scanner(new File ("student.txt"));
        while(input.hasNext()){
          Student stu = new Student(input.next(), input.nextInt(), input.next(), input.nextInt());
          stuList.put(stu.getId(), stu);
        }
        input.close();
      }
      catch(FileNotFoundException e){
        System.out.println("Do not find file.");
      }
    }
```

signIn method:

- By clicking “Enter” button or the return key on the keyboard will allow a method in Sign class called signIn to sign in students.

```java
public void signIn(int studentId){
  boolean valid=false;
    for(Integer id : stuList.keySet()){
      if(studentId==id){
        Student a= stuList.get(id);
        a.setLate(false);
        valid=true;
      }
    }
  //check whether the id number input is valid
  if(valid){
      valid=false;
  }
  else{
    JFrame frame=new JFrame("Invalid ID Number");
    JLabel label=new JLabel("The ID number you enter does not exist. \n Please enter again.");
    frame.add(label);
    frame.setSize(400,200);
    frame.setDefaultCloseOperation(frame.DISPOSE_ON_CLOSE);
    frame.setVisible(true);
  }
}
```

- The method catches error entry from the user by prompting a new window saying the number entered is invalid.



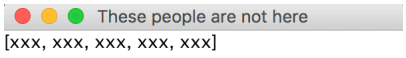
endSign method:

- it is called when the user clicks “End Sign In” button in Sign In Window, so a lateList is made and stored as an instance variable in Sign class

```
public void endSign(){
  for (Integer id : stuList.keySet()){
      Student a= stuList.get(id);
      if(a.getLate()==true){
        lateList.add(a.getName());
        a.addLateT();
      }
      else {
        a.setLate(true);
      }
  }
}
```

getLateList method:

- Then, SignFrame use getLateList method in Sign class to display the late list in a new window as following



If the user clicks any “Exit” button, it will bring the user to the previous displayed window.

Student class:

- I created a Student class so that the user is able to get and modify student information stored in the Student object with its method

```
public class Student{

  private String name;
  //student name
  private int id;
  //student ID number
  private String tieColor;
  //student tie color
  private int lateT;
  //Total times the student has been late
  private boolean late;
  //whether the student is currently late
  /*
    Constructor: to input student name, ID number, and tie color ahead
  */
  public Student(String tieColor, int id, String name, int lateT){
  public void setName(String name){
  //This method is a setter to change the late status of the student
  public void setLate(boolean status){

  public void addLateT(){

  public void takeOffT(){

  public String getName(){

  public int getId(){

  public String getTie(){

  public int getLateTime(){

  public boolean getLate(){
}
```

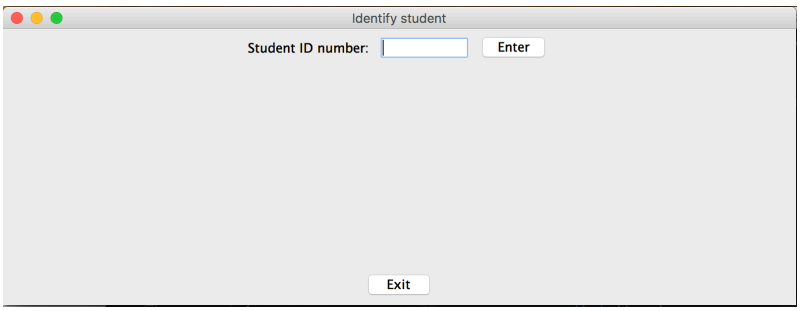
setLate method:

- Let the user to change the late status of the student with the correlated ID number to fasle (meaning not late)

```
  public void setLate(boolean status){
    late=status;
  }
```

Modify Window:

- When the user clicks the “Modify student information” button, Identify class will be declared to display the following new interface window for users to input the student ID number to identify which student’s information they wish to change.



Identify class:

- Sets up what the interface should look like

```
public class Identify extends JFrame{
  private static JButton enter_btn=new JButton("Enter");
  private static JButton exit_btn=new JButton("Exit");
  private static JTextField sid_text= new JTextField(7);
  private static JLabel sid_label=new JLabel("Student ID number: ");
  private Student student;
  private Sign sign;

  public Identify(Sign sign){
    this.sign=sign;
    createInteractors();
    setTitle("Identify student");
    setSize(800,300);
    setDefaultCloseOperation(EXIT_ON_CLOSE);
  }
```

createInteractor method:

- Add the components into the window frame

```
public void createInteractors(){
  MyListener listen= new MyListener();

  Panel pnl1=new Panel();
  Panel pnl2=new Panel();
  add(pnl1, BorderLayout.CENTER);
  add(pnl2, BorderLayout.SOUTH);

  pnl1.add(sid_label);

  sid_text.setDocument(new JTextFieldLimit(7));
  sid_text.setActionCommand("Enter");
  sid_text.addActionListener(listen);
  pnl1.add(sid_text);

  enter_btn.addActionListener(listen);
  pnl1.add(enter_btn);

  exit_btn.addActionListener(listen);
  pnl2.add(exit_btn);
}
```

MyListener class:

- Allows the buttons to have meaning so that depending on the button that the user has clicked different method will be called.

```
public class MyListener implements ActionListener{

  @Override
  public void actionPerformed(ActionEvent e){
    //when click enter or pressed "return"
    if(e.getSource()==enter_btn || e.getSource()==sid_text){
      //get the input ID nnumber in the text field
      int id=Integer.parseInt(sid_text.getText());
      //sign in
      student=sign.getStudent(id);
      if(student!=null){
        dispose();
        ModifyFrame modifyFrame=new ModifyFrame(student, sign);
        modifyFrame.setVisible(true);
      }
    }
    else if(e.getSource()==exit_btn){
      dispose();
      MainFrame mainFrame=new MainFrame();
    }
  }
}
```

JTextFieldLimit class:

- extends from PlainDocument class to prevent user from inputting more than seven digits number (as client has requested)

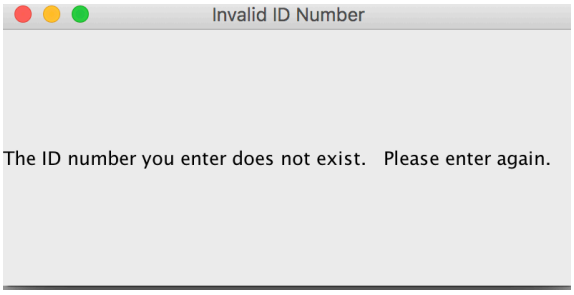
```
public class JTextFieldLimit extends PlainDocument {
  private int limit;
  JTextFieldLimit(int limit) {
    super();
    this.limit = limit;
  }
  @Override
  public void insertString( int offset, String  str, AttributeSet attr) throws BadLocationException{
    if (str == null) return;

    if ((getLength() + str.length()) <= limit) {
      super.insertString(offset, str, attr);
    }
  }
}
```

getStudent method:

- After inputting the student number, getStudent method in Sign class will catch error entry if the user types in a number that is not valid.

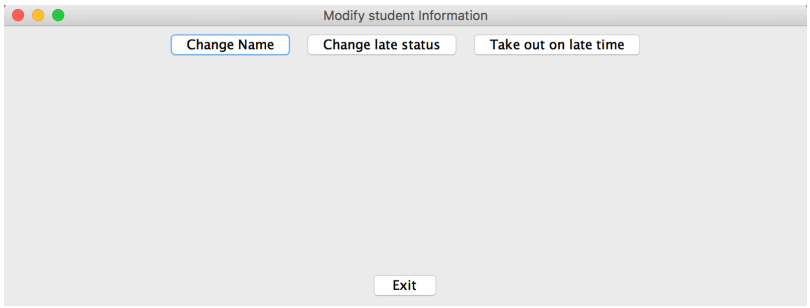
```
public Student getStudent(int studentId){
  for(Integer id : stuList.keySet()){
    if(studentId==id){
      Student a= stuList.get(id);
      return a;
    }
  }
  JFrame frame=new JFrame("Invalid ID Number");
  JLabel label=new JLabel("The ID number you enter does not exist. \n Please enter again.");
  frame.add(label);
  frame.setSize(400,200);
  frame.setDefaultCloseOperation(frame.DISPOSE_ON_CLOSE);
  frame.setVisible(true);
  return null;
}
```

- When an error entry is inputted, it will show as following:



Modify Window:

After entering the valid number, a new window will display to ask the user what to change by creating a ModifyFrame class:



ModifyFrame class:

- Sets up the interface window

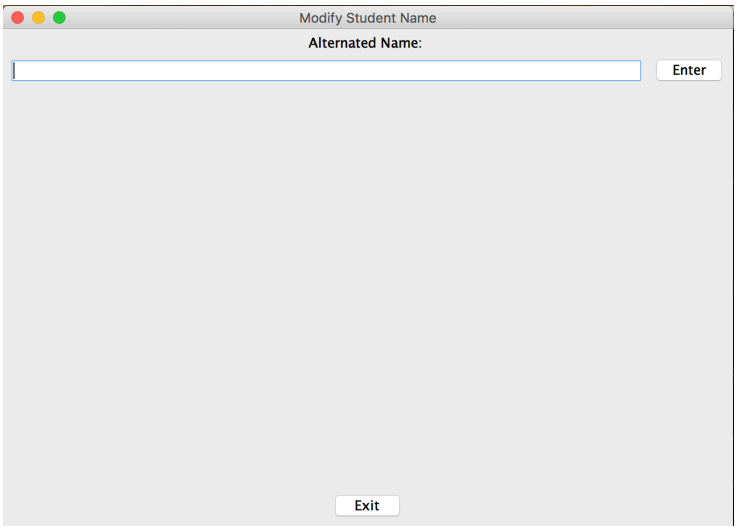
```
public class ModifyFrame extends JFrame{
  private static JButton name_btn= new JButton("Change Name");
  private static JButton status_btn= new JButton("Change late status");
  private static JButton time_btn= new JButton("Take out on late time");
  private static JButton exit_btn=new JButton("Exit");
  private Student student;
  private Sign sign;


  public ModifyFrame(Student student, Sign sign){
    this.student=student;
    this.sign=sign;
    createInteractors();
    setTitle("Modify student Information");
    setSize(800,300);
    setDefaultCloseOperation(EXIT_ON_CLOSE);
  }

  public void createInteractors(){
      MyListener listen= new MyListener();

      Panel pnl1=new Panel();
      Panel pnl2=new Panel();
      Panel pnl3=new Panel();
      add(pnl1, BorderLayout.CENTER);
      add(pnl2, BorderLayout.SOUTH);

      name_btn.addActionListener(listen);
      pnl1.add(name_btn);

      status_btn.addActionListener(listen);
      pnl1.add(status_btn);

      time btn.addActionListener(listen);
      pnl1.add(time_btn);

      exit_btn.addActionListener(listen);
      pnl2.add(exit_btn);
```

MyListener class:

- Allows the buttons to have meaning so that depending on the button that the user has clicked different method will be called.

```java
public class MyListener implements ActionListener{
  private JFrame frame=new JFrame();
  private JLabel label;
  @Override
  public void actionPerformed(ActionEvent e){
    if(e.getSource()==name_btn){
      dispose();
      NameFrame nameFrame=new NameFrame(student, sign);
      nameFrame.setVisible(true);
    }
    else if(e.getSource()==status_btn){
      dispose();
      StatusFrame statusFrame=new StatusFrame(student, sign);
      statusFrame.setVisible(true);
    }
    else if(e.getSource()==time_btn){
      dispose();
      frame=new JFrame();
      if(student.getLateTime()>0){
        student.takeOffT();
        label=new JLabel("You have changed "+student.getName()+" has a total late time of "+student.getLat

      }
      else{
        label=new JLabel("The student is never late, so the operation is invalid.");
      }
      frame.add(label);
      frame.add(exit_btn, BorderLayout.SOUTH);
      frame.setSize(400,200);
      frame.setDefaultCloseOperation(frame.DISPOSE_ON_CLOSE);
      frame.setVisible(true);
    }
```

If the user clicks “Change name,” the following window will be displayed by calling NameFrame class



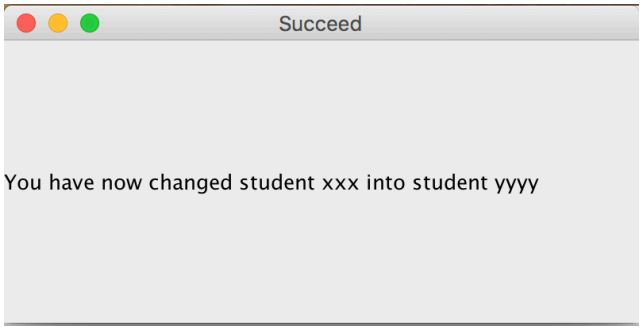
NameFrame class:

- When the user enters the new name, it will store the change to the student’s Student object and display another window to show the change made

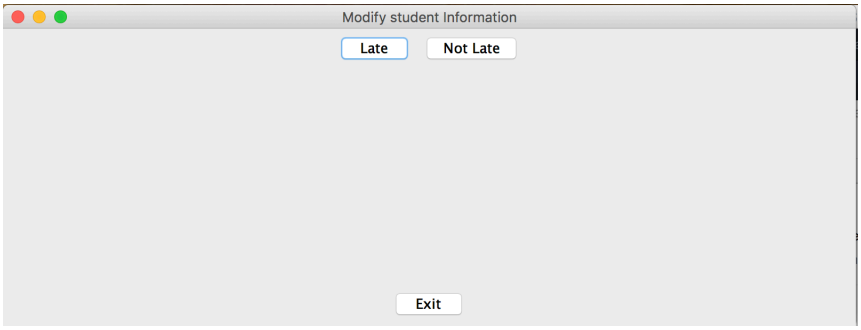
```
public class NameFrame extends JFrame{
    private static JButton enter_btn=new JButton("Enter");
    private static JButton exit_btn=new JButton("Exit");
    private static JTextField sid_text= new JTextField(50);
    private static JLabel sid_label=new JLabel("Alternated Name: ");
    private Sign sign;
    private Student student;

    public NameFrame(Student student, Sign sign){
      this.sign=sign;
      this.student=student;
      createInteractors();
      setTitle("Modify Student Name");
      setSize(700,500);
      setDefaultCloseOperation(EXIT_ON_CLOSE);
    }

    public void createInteractors(){
      MyListener listen= new MyListener();

      Panel pnl1=new Panel();
      Panel pnl2=new Panel();
      add(pnl1, BorderLayout.CENTER);
      add(pnl2, BorderLayout.SOUTH);

      pnl1.add(sid_label);
```

```
      sid_text.setActionCommand("Enter");
      sid_text.addActionListener(listen);
      pnl1.add(sid_text);

      enter_btn.addActionListener(listen);
      pnl1.add(enter_btn);

      exit_btn.addActionListener(listen);
      pnl2.add(exit_btn);
    }
    public class MyListener implements ActionListener{
      @Override
      public void actionPerformed(ActionEvent e){
        if(e.getSource()==enter_btn || e.getSource()==sid_text){
          String old_name=student.getName();
          String name=sid_text.getText();
          student.setName(name);
          JFrame frame=new JFrame("Succeed");
          JLabel label=new JLabel("You have now changed student "+ old_name+ " into student"+ name);
          frame.add(label);
          frame.setSize(400,200);
          frame.setDefaultCloseOperation(frame.DISPOSE_ON_CLOSE);
          frame.setVisible(true);

        }
        else if(e.getSource()==exit_btn){
          dispose();
          ModifyFrame modifyFrame=new ModifyFrame(student, sign);
          modifyFrame.setVisible(true);
        }
      }
    }
}
```

- If I type “yyyy” in the text field, I change “xxx”(default name in my program) and this following window will be shown meaning the change is successfully done.



If the user clicks “Change late status” button, StatusFrame class will be declared to create a new interface window as following:

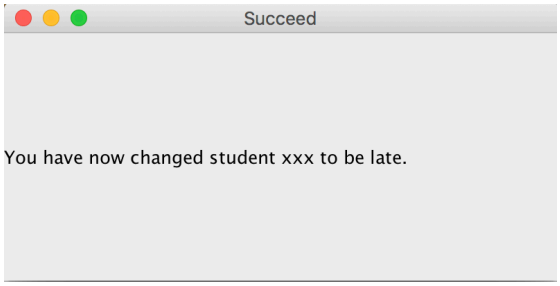


StatusFrame class:

- Set up a new interface window to let the user to control what to do

```
public class StatusFrame extends JFrame{
  private JButton true_btn =new JButton("Late");
  private JButton false_btn =new JButton("Not Late");
  private JButton exit_btn=new JButton("Exit");
  private Student student;
  private Sign sign;

  public StatusFrame(Student student, Sign sign){
    this.student=student;
    this.sign=sign;
    createInteractors();
    setTitle("Modify student Information");
    setSize(800,300);
    setDefaultCloseOperation(EXIT_ON_CLOSE);
  }
  public void createInteractors(){
    MyListener listen= new MyListener();

    Panel pnl1=new Panel();
    Panel pnl2=new Panel();
    add(pnl1, BorderLayout.CENTER);
    add(pnl2, BorderLayout.SOUTH);



    true_btn.addActionListener(listen);
    pnl1.add(true_btn);

    false_btn.addActionListener(listen);
    pnl1.add(false_btn);

    exit_btn.addActionListener(listen);
    pnl2.add(exit_btn);
```

- Will give different window that display the confirmation of the change the user made
    - If click “Late”



    - If click “Not Late”

MyListener class:

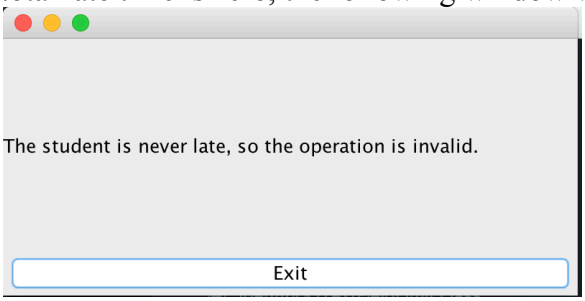
- Allows the buttons to have meaning so that depending on the button that the user has clicked different method will be called.

```
public class MyListener implements ActionListener{
  private JLabel label;
  public void actionPerformed(ActionEvent e){
    JFrame frame=new JFrame("Succeed");
    if(e.getSource()==true_btn || e.getSource()==false_btn){
      if(e.getSource()==true_btn){
        student.setLate(true);
        label=new JLabel("You have now changed student "+ student.getName()+ " to be late.");
      }
      else{
        student.setLate(false);
        label=new JLabel("You have now changed student "+ student.getName()+ " to be here on time.");
      }
      frame.add(label);
      frame.setSize(400,200);
      frame.setDefaultCloseOperation(frame.DISPOSE_ON_CLOSE);
      frame.setVisible(true);
    }
    else if(e.getSource()==exit_btn){
      dispose();
      ModifyFrame modifyFrame=new ModifyFrame(student, sign);
      modifyFrame.setVisible(true);
    }
  }

}
```

If the user clicks “Take out one late time” button, the student’s late time will be subtract by one by using the takeoff method in Student class

```
public void addLateT(){

public void takeOffT(){
  lateT--;
}
```

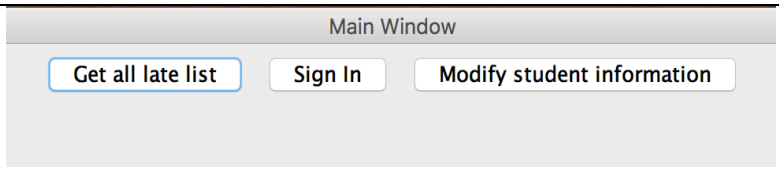
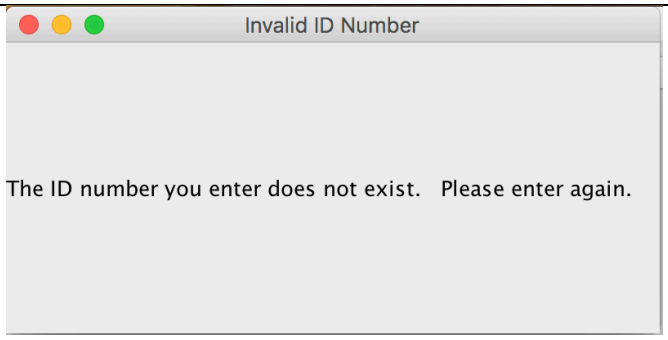
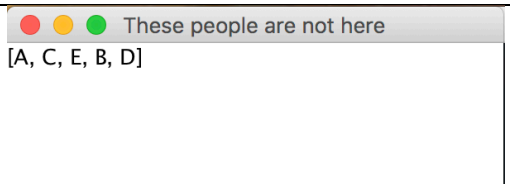
However, if the student total late time is zero, the following window will appear

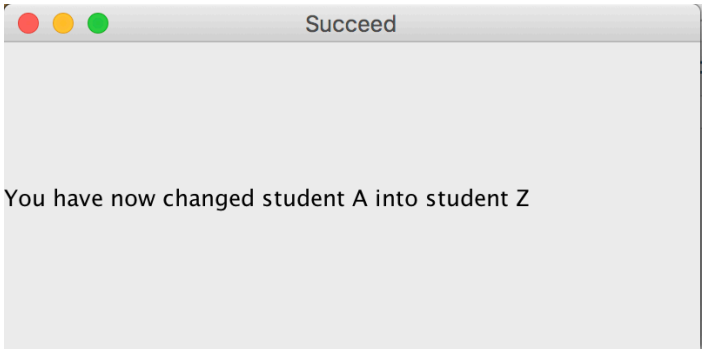


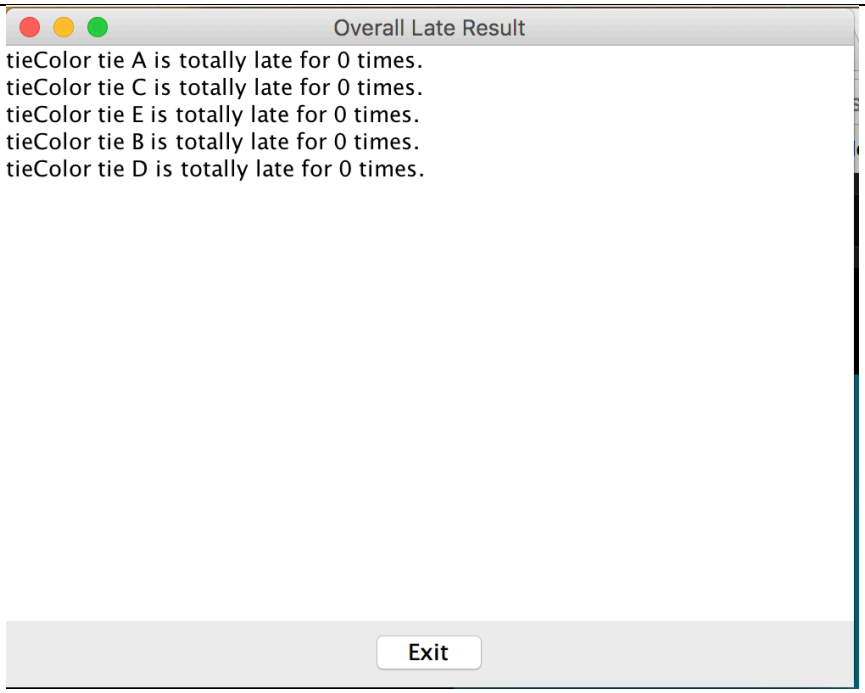
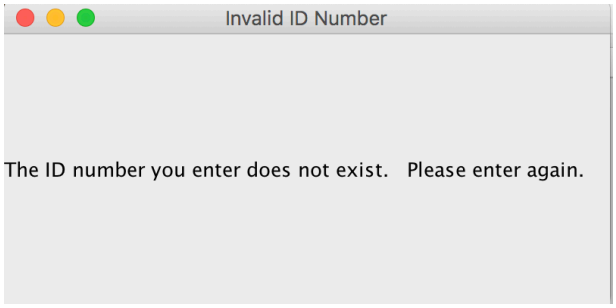
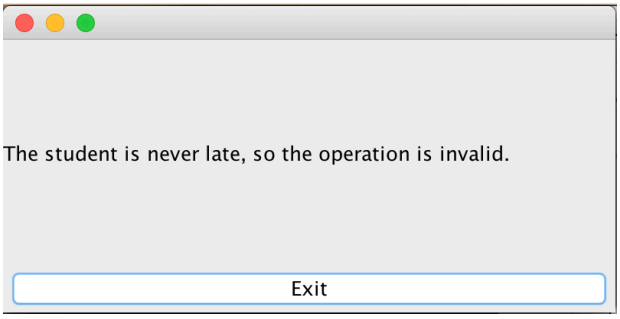
If the users want to end the program, they have to click on the exit button on the tool bar. The “Exit” button I creates will only allow user to go back to the previous window they are at.

Criterion E: Evaluation

Success criteria

The program should start with three options for the user to pick: sign in, get all late list, and modify student information		Accomplished
The program allows students to sign in by inputting student ID number	When a valid number is inputted, the corresponded student name does not appear in the late list window when sign in is ended	Accomplished
If an invalid number is inputted, info message will appear on the screen		Accomplished
When end sign in, the list of students who are absent will appear in a new window		Accomplished (though the client has not requested the students to be listed in order, the order of the students is randomly arranged because I use a HashMap class for lists)

The program allows users to modify student's information	A confirmation window will appear to make sure the change is indeed made in the student information that is store in Student class 	Accomplished
Can share to program with other dorm parents in different lap top who are on duty that day	Everyone can use this software as they have the file and a platform to run java.	Accomplished
There will be a student data base created by teacher to input as the information hold in Student class before the program starts	A student.txt is created so dorm parents can first type in every student data base	Accomplished

The program generates an easy-access and digital students' accumulation late time		Accomplised
In case of data entry error, info message will appear on the screen	If enter invalid student number:  If asks to take off one late time, while the student is never late 	Accomplished

Recommendations for Further Development

- Password for dorm parent's section: adding unique login and password so that only dorm parents can access to modify student information
- More features: maybe add another class that informs the student is excused out of school to have dinner or to go to mall. Enabling dorm parents to track down for students who are missing but are excused.
- The worst student: maybe add a method to detect whether the student's total late time exceeds a certain number. And when they do so, dorm parents will be informed and give the student a little punishment.
- Design: maybe devote more time on how the interface window should look like. Add up to its simple and easiness, make it more embellished.