

Table of Contents Page #

[Stage A – Analysis 3](#_Toc350465827)

[A1 Analysing the Problem 3](#_Toc350465828)

[A2 Criteria for Success 8](#_Toc350465829)

[A3: Prototype Solution 9](#_Toc350465830)

[Stage B – Detailed Design 17](#_Toc350465831)

[B1 Data Structures: 17](#_Toc350465832)

[B3: Modular Organization 32](#_Toc350465833)

[Stage C – Program 34](#_Toc350465834)

[C1 Using good programming style 34](#_Toc350465835)

[C2 Handling Errors 96](#_Toc350465836)

[C3 Success of the Program 97](#_Toc350465837)

[Stage D - Documentation 99](#_Toc350465838)

[D1 Including an annotated hard copy of the test output 99](#_Toc350465839)

[D2 Evaluating Solutions: 106](#_Toc350465840)

--------------------------

Stage A – Analysis

Peng Fei Wang

----------------------------

## Stage A – Analysis

# A1 Analysing the Problem

**Description of Problem**

In order to obtain an Ontario Secondary School Diploma (OSSD), students must complete a variety of courses through their 4 years in high school. Classes are structured into 75 minute blocks, where during this time the teacher will teach students material necessary to obtain the credit for the course. In order for students to apply the knowledge learned in class they are given homework. Practice questions to be done at home to solidify their understanding of the material taught. In order to quantify how well students are learning in the class, students are given evaluations in the form of quizzes, tests and assignments. All of which will be used to formulate a course mark to be included in each students transcript. The course marks are used by universities and colleges to judge whether or not a student will be accepted to their particular program. As a result completing homework and preparing for evaluations are essential for a student to succeed in high school. This becomes difficult when a student will have several different classes in a day and must remember information such as due date, homework questions, evaluation dates, etc.

With a high school student as a client, my goal is to create an organiser that will help students keep track of their homework, quiz, test, and assignments. The student would be able to enter this information into specific days with the ability to go back to read what was recorded on each particular day. With prompts available for quizzes, tests, and assignments due within a span of 7 days.

**Analysis of Problem**

After meeting with my client, the functions of the program were discussed. After deciding the client would like an electronic agenda made in order to reduce paper usage, and extra space.

Transcript of Interview:

Q: Should tasks be assigned to the date assigned or due date?

A: the tasks should be assigned to the date assigned but still link the due date to the task.

Q: What Extra functions would you like in the program?

A: I would like the various tasks to be split into categories. With each task connected to a due date, and have to program create a list based on the due date so I can prioritize what tasks I must complete.

Q: How would you like a list of tasks to be sorted?

A: Both by due date and by date assigned.

Q: What categories would you like the task to be split into?

A: By homework, assignments, quiz, and test.

**Current Systems**

The most widely used solution to organising school related activities is the use of specially formatted notebooks named agendas with rectangular boxes under each school day for students to jot down information. These agendas are sold by the school to students in the start of every school year. However due to causes such as reduction of paper use, the agendas sold are becoming very small (each box is only 5 by 8 centimetre squared). This combined with increased amounts of homework, quizzes, tests, and assignments as students progressed in school, leads to insignificant space in the agenda.

To keep track of these school related activities, students rely on agendas provided for sale by the school in the form of a palm sized notebook. In order to reduce paper usage, the sizes of agendas have shrink resulting in each daily section shrinking to only 5\*8cm box. This in conjunction to the increase in homework load and evaluations as students move onto higher grades such as grade 12, there simply isn’t enough space to record everything students believe to be important in the space provided.

**Input, Output:**

The input required by the program include: the date, the day the user wish to add homework/quiz/test/assignment to, information concerning the homework/quiz/test/assignment in question such as due date. Input data always either allows the user to go back or asks for confirmation of inputs. If that fails the user can always delete what was inputted and re-input the data.

There are several outputs associated with this program. The program can output what the user has previously stored in the program, a list of upcoming evaluations in the next 7 days, a list of homework due in the next seven days or a list of evaluations and homework that occurs or is due that day.

**Systematic analysis**

**Figure A1- Data Flow Diagram for Reading and Writing to file**

Add information/ Read Previous Information/ Quick Check/ Search for school work/activity relative to due date/ Check off finish schoolwork

/

Day data file

The user selects date to access

Rewrite the Day Data file after new information is added or old information is deleted

**Figure A2 – System Flow Chart for adding information**

Prompts user to enter information

User enters selection for adding information

User confirms information is entered correctly

User enters relevant information

Updated Day data file

Temporary Storage in Memory

**Figure A3 – System Flow Chart for Searching for school work/activities**

Prompts the user for the due date of the school work/activities

User selects option to search for school work/activities

Display search result

Search for all school work/activities due that day

User enters due date

Temporary storage in memory

**Figure A4 System flow chart for displaying data**

Display relevant output

Find relevant Data

User enters input to request for output

Temporary Storage in memory

**Figure A5 System flow chart for sorting school work/activities**

User selects option to sort school work/activities

Sort the list of school work/activities according to option selected

Update the Day data file

User indicates they are finished with using the sorted list

Temporary storage in memory

# A2 Criteria for Success

**Outline of Proposed Solution**

1. Store information on school work/activity in the day the school work/activity was assigned.
2. Allow user to access information entered above
3. Allow user to check off finished school work
4. Allow user to quickly check what evaluations and school work that are due in the next 7 days
5. Store all information in data files.

**Limitations/Requirements**

**Requirements:**

- JRE(version 6)

- Windows XP/Vista/7/8

-Input device (keyboard)

-Output device (LCD monitor)

**Restrictions:**

* Calendar is restricted to 2013
* The user must know the current date
* Once a school work is marked finished it cannot be unmarked
* Information entered must be categorized as either homework, quiz, test, or assignment

**Goals:**

* User can input as much information as needed for each school work/activity
* Information can be deleted
* Information can be viewed at a later time
* Information can be sorted relative to both date assigned and date due.
* Information can be searched relative to due date
* Homework and assignments can be marked as complete
* List of school work/activities due in the next 7 days can be generated

**Objectives:**

* Have indicators such as “success” when an action is performed
* When inappropriate data is inputted the program will re-prompt the user for appropriate data.
* Inappropriate input by user will not crash the program
* Actions aside from exit will always loop back to the main menu.

# A3: Prototype Solution

Ask the user for confirmation that all the information is entered correctly

Ask the user which school work from homework/quiz/test/assignment they want to view

Ask the user if they would like to view homework, quiz, test, or assignment

Ask the user which day they would like to revisit

Read Previous Information

Ask the user to enter anymore additional information concerning the school work

Asks the user for the due date

Ask the user if its homework, quiz, test or assignment

Add Information

Menu

Asks user for Date

Display stored information with options to mark as complete, to delete, and to return to main menu

Return to main menu

Exit

Ask the user for the due data

Search for school work/activity with respect to due date

Quick Check

Ask the user if they would like to see the list sorted with ascending due dates or ascending days past.

Ask the user if they wish to view homework/quiz/test or all due/occurring in the next week

Display all tasks due on that day with options to mark as complete, to delete, and to return to main menu

Display result with options to mark as complete, to delete, and to return to main menu

**Description:**

This is the first page the user will be introduced to. Here the user will input the current month using the numbers 1-12 each representing a separate month. Invalid inputs (ie string, numbers other than 1-12, etc) will lead to a re-prompt for appropriate data.

Please Select the Current Month:

1. January
2. February
3. March
4. April
5. May
6. June
7. July
8. August
9. September
10. October
11. November
12. December

Please enter your choice: 11

**Description:**

This is the main menu where the user will select which activity they would like to select. The numbers 1-6 will be accepted as input for each respective action, with inappropriate input resulting in re-prompts.

Main Menu

--------------------------------

1. Add information
2. View Information
3. Quick Check
4. Search for work/activities with respect to due date
5. Check off completed school work
6. Exit

Please Enter an Option: 1

Please Enter the Present Day of the Month: 18

**Description:**

In this page the user will enter the day of the month with numbers 1-31. Inappropriate inputs will lead to re-prompts.

Add Information:

Which category does the new school work/activity fall under?

1. Homework
2. Assignment
3. Quiz
4. Test

Please enter your choice: 1

**Description:**

In option 1, the user will be greeted first by a categorizer page. Where the user must select which category the new school work/activity falls under. Using the numbers 1-4 to signify choices, with inappropriate inputs leading to re-prompts.

**Description**:

After the category has been established options will be presented regarding the due date. Numbers from 1-3 will be used to signify the choices, with inappropriate inputs leading to re-prompts. Choice number 3 will result in pages similar to the first 2 which will ask the user to specify the appropriate due date.

New Homework

Is the homework due?

1. Tomorrow
2. The day after tomorrow
3. Other

Please select the appropriate choice: 1

**Description**:

Here the user will enter all the information they feel is relevant to the homework.

Please enter information concerning this homework:

Page 218 #2-8 Math

Confirm new Homework:

Due Date: November 19

Page 218 #2-8 Math

Confirm entry? (y/n) y

**Description:**

Here the user is asked for confirmation of what was entered. The letters y would denote yes and n would denote no. Answer of yes would result in the new homework being recorded while an answer of no will result in the entry being disregarded. Inappropriate input will result in a re-prompt. Confirmation will to lead to return to main menu

**Description:**

Here the user is asked which category they would like to view.

Which category do you wish to view?

1. Homework
2. Assignment
3. Quiz
4. Test
5. All

Please enter your choice: 1

**Description:**

When the user choices option 2 they are greeted with this page. Where the user is asked which page they would like to revisit

Which Day would you like to revisit?

1. Today
2. Yesterday
3. The day before Yesterday
4. Other

Please enter your choice: 1

Quick Check:

1. All homework due in the next 7 days
2. All assignments due in the next 7 days
3. All quizzes occurring in the next 7 days
4. All tests occurring in the next 7 days
5. Everything due/occurring in the next 7 days

Please select one of the options above: 1

Homework on November 18 2013

1. Page 218 #2-8 Math Due November 19
2. Mark all as Finished
3. Delete all
4. Return to Main Menu

Please select one of the options above: 4

**Description:**

Here the user is able to see all the information concerning each homework that they have entered before. The user can select a homework to mark it off as complete or to delete it, or the user can return to the main menu.

**Description:**

In option 3 the user is able to check for school work/evaluations due/happening in the next 7 days. The user first must choose which category they would like to view, using the numbers 1-5 to signify the choices.

Homework due in the next 7 days:

1. Page 218 #2-8 Math Due November 19Unfinished
2. Return to Main Menu

Please select one of the options above: 2

**Description:**

The user is presented with all the homework due in the next 7 days. With the words Finished next to homework already checked off and Unfinished for homework that’s not marked as complete. The user is able to select a homework to mark as complete or delete it, or the user can return to the main menu.

User comment: Can the Finished and Unfinished homework and assignments be placed into separate categories?

**Description:**

The user is presented with all the school work/activities due on the day selected. Homework and assignments will be marked as Finished or Unfinished depending on completion. The user is able to select homework to mark as complete or to delete it and the option to return to the main menu.

Search for school work/activity in respect to due date:

Which day is the school work/activity due?

1. Tomorrow
2. The day after tomorrow
3. Other

Please select one of the options above: 1

School work/activity due on November 19:

1. Page 12 #2-8 Math Unfinished
2. Return to Main Menu

Please select on the options above: 2

**Description:**

When option 3 is selected the user is asked for the day that the school work/activity is due. When option 3 is selected the program will bring to user to 2 pages similar to the first 2 pages of the program to ask for a specific date.

User comment:

Can the school work/activity be split into categories like the actions?

**Revisions:**

* Have the finished and unfinished homework and assignments outputted under separate titles
* Have options to split the output of searching by due date by the categories: homework, quiz, test, and assignment.

--------------------------

Stage B – Detailed Design

Peng Fei Wang

----------------------------

## ICS 4U7 Dossier Stage B – Detailed Design

# B1 Data Structures:

**Storing Information for an individual day**

A class called Day is created to store the information concerning assignments stored previously by the user. Different variables will be used to store information concerning homework, quiz, test, assignment, the corresponding due date and the date assigned. All information concerning a particular day will be stored in this object.

public class Day

{

private String [][] homework = new String [10][2];

private String [][] quiz = new String[4][2];

private String [][] test = new String[4][2];

private String [][] assignment = new String[4][2];

private int [][] duedatehomework = new int[10][2];

private int [][] duedatequiz = new int[4][2];

private int [][] duedatetest = new int[4][2];

private int [][] duedateassignment = new int[4][2];

private int [][] assigneddate = new int[2];

methods…

}

Strings are used for homework, quiz, test, and assignment so the user is able to input any information that they feel would be beneficial to reminding them of what occurred the availability of letters is essential for this to occur. They are 2 dimensional arrays because each homework, quiz, test, and assignment will also need to be identified as finished or not finished. The due dates and assigned dates for each particular school work are assigned to 2 dimensional integer arrays with a size of [x][2] because while both month and day are mutually inclusive to date they are mutually exclusive of each other so by separating them in this fashion the date can be easily determined. As opposed to using only a single integer, since it would require more complicated processes to separate a date entered in formats such as dd/mm, or recording only the days past starting from January first as opposed to just separating the month and day into distinct integers. They are organized into 2 dimensional arrays to correspond to the various number of homework/quiz/test/assignment. The date must be in the form of a number as opposed to a String because the dates are compared mathematically for certain functions so it would be illogical to store the information in String only to have to covert them again in the program.

**Storing Multiple Days:**

The Day objects are grouped according to which month the day object resides, 3 different classes will be used to differentiate between 28, 30, and 31 day months. The 3 classes will be named TwentyEightDayList, ThirtyDayList, and ThirtyOneDayList. The Day objects will be stored in an array in ascending order of day. This class will also be responsible for inserting each Day object to the appropriate location in the array after the information is obtained from the save file and to return each specific Day object. 12 of these classes will be initialized to encompass the 12 months of the year which will include all the Day objects for the year. These classes will be none static since multiple months will share the same class to store Day objects.

public class TwentyEightDayList

{

private static final int ListLength = 28;

private Day [] daylist = new Day [ListLength];

methods…

}

1D Array of Day objects:

|  |  |
| --- | --- |
| 00 | Day 1 location |
| 01 | Day 2 location |
| 02 | Day 3 location |
|  |  |
| 26 | Day 27 location |
| 27 | Day 28 location |

A 1 dimensional array is used to store the Day objects grouped into each month because the number of Day object in each month is constant so dynamic lists such as linked lists would not be required but the number of days of each month is different from each other so a 2 dimensional array will have empty references that are not desired. While having 12 different classes to group the days it will still be relatively simple to differentiate between the months.

\*Since the number of Days are constant Day objects cannot be deleted or added to the array, the contents of the Day objects such as test, homework, etc can be set to null but the Day object will always exist in the 1 dimensional array.

**Storing Day objects that are within 7 days of the current date**

In order to display to the user homework/test/quiz/assignment due in the next 7 days a list must be made to contain the individual Day objects that meet the previous stated criteria. This list must be dynamic since the number of Day objects that will meet the criteria is not constant and will be different from user to user a dynamic storage system is needed. The class DayNode is an object that operates as a node with a Day object acting as the information with a next variable linking to the next DayNode with a different Day object that also fits the criteria. The DayNode objects are linked in the DayQueue class as a linked list that could be sorted and displayed to the user.

public class DayNode

{

Day info;

DayNode link;

}

public class DayQueue

{

DayNode head;

}

DayNode

head

DayNode

Day

link

Day

link

DayNode

Day

link

**......**

DayNode

**Reading and Writing to Save File**

All the information entered by the user and stored in a Day object will be stored in an external storage text file to preserve the data for future use. The data will be saved when the user exists the program into a text file. When the program is initialized 365 Day objects will be initialized and the corresponding information from the text files will be transferred to the various Day objects.

Saved File Format:

|  |
| --- |
| <Assigned Month>  <Assigned Day>  <Homework 1>  <State of Homework1>  <Homework 1 Due Month>  <Homework 1 Due Day>  <Homework 2>  <State of Homework2>  <Homework 2 Due Month>  <Homework 2 Due Day>  <Homework 3>  <State of Homework3>  <Homework 3 Due Month>  <Homework 3 Due Day>  <Homework 4>  <State of Homework4>  <Homework 4 Due Month>  <Homework 4 Due Day>  <Homework 5>  <State of Homework5>  <Homework 5 Due Month>  <Homework 5 Due Day>  <Homework 6>  <State of Homework6>  <Homework 6 Due Month>  <Homework 6 Due Day>  <Homework 7>  <State of Homework7>  <Homework 7 Due Month>  <Homework 7 Due Day>  <Homework 8>  <State of Homework8>  <Homework 8 Due Month>  <Homework 8 Due Day>  <Homework 9>  <State of Homework9>  <Homework 9 Due Month>  <Homework 9 Due Day>  <Homework 10>  <State of Homework10>  <Homework 10 Due Month>  <Homework 10 Due Day>  <Assignment 1>  <State of Assignment 1>  <Assignment 1 Due Month>  <Assignment 1 Due Day>  <Assignment 2>  <State of Assignment 2>  <Assignment 2 Due Month>  <Assignment 2 Due Day>  <Assignment 3>  <State of Assignment 3>  <Assignment 3 Due Month>  <Assignment 3 Due Day>  <Assignment 4>  <State of Assignment 4>  <Assignment 4 Due Month>  <Assignment 4 Due Day>  <Quiz 1>  <State of Quiz 1>  <Quiz 1 Due Month>  <Quiz 1 Due Day>  <Quiz 2>  <State of Quiz 2>  <Quiz 2 Due Month>  <Quiz 2 Due Day>  <Quiz 3>  <State of Quiz 3>  <Quiz 3 Due Month>  <Quiz 3 Due Day>  <Quiz 4>  <State of Quiz 4>  <Quiz 4 Due Month>  <Quiz 4 Due Day>  <Test 1>  <State of Test 1>  <Test 1 Due Month>  <Test 1 Due Day>  <Test 2>  <State of Test 2>  <Test 2 Due Month>  <Test 2 Due Day>  <Test 3>  <State of Test 3>  <Test 3 Due Month>  <Test 3 Due Day>  <Test 4>  <State of Test 4>  <Test 4 Due Month>  <Test 4 Due Day> |

Each Day object’s data will be stored in the above format; no entries will result in a blank line. Thus each Day object will fill a constant number of lines; this will be how the program will distinguish each Day object saved in the same file. Due to the constant number of days in a month the amount of lines will be a constant value, when this value has been reached the program will know that the program has ended. Since the data is only updated when the user exists the program, exiting with the use of means other than the exit option in the program will result in loss of data.

**B2 Algorithms**

**Day class**

Accessor:

getHomeWork(int index) return String

getHomeWorkState(int index) return String

getQuiz(int index) return String

getQuizState(int index) return String

getTest(int index) return String

getQuizState(int index) return String

getAssignement(int index) return String

getAssignementState(int index) return String

getDueDateHomeWork(int index, int dateindex) return int

getDueDateQuiz(int index, int dateindex) return int

getDueDateTest(int index, int dateindex) return int

getDueDateAssignment(int index, int dateindex) return int

getAssignedDate(int dateindex) return int

Mutator:

setHomeWork(int index, String input)

setHomeWorkState(int index, String input)

setQuiz(int index, String input)

setQuizState(int index, String input)

setTest(int index, String input)

setTestState(int index, String input)

setAssignement(int index)

setAssignementState(int index, String input)

setDueDateHomeWork(int index, int duemonth, int dueday, String input)

setDueDateQuiz(int index, int duemonth, int dueday, String input)

setDueDateTest(int index, int duemonth, int dueday, String input)

setDueDateAssignment(int index, int duemonth, int dueday, String input)

setAssignedDate(int dateindex, String input)

toStringHomework(int index)

toStringQuiz(int index)

toStringTest(int index)

toStringAssignement(int index)

\*The information below applies to all toString methods, differing types of schoolwork will have different variables but the basic function is the same. The example below is for the toStringHomeWork(int index) function.

Description: Outputs information concerning a specific schoolwork including date due and completion.

Parameter: int index

Return type: void

Preconditions: Information concerning the school must have been previously inputted by user.

Post conditions:

Algorithm:

Print (this.getHomeWork(index) + this.getDueDateHomeWork(index, 0)+ this.getDueDateHomeWork(index, 1) + this.getHomeWorkStatus(index))

**TwentyEightDayList Class**

Accessor:

getDayList(int index) return Day

Mutator:

setDayList(int index, Day newDay)

**ThirtyDayList Class**

Accessor:

getDayList(int index) return Day

Mutator:

setDayList(int index, Day newDay)

**ThirtyOneDayList Class**

Accessor:

getDayList(int index) return Day

Mutator:

setDayList(int index, Day newDay)

**DayNode Class**

Accessor:

getinfo() return Day

getLink() return DayNode

Mutator:

setinfo(Day day)

setLink(DayNode newnode)

**DayQueue Class**

**Append End**

Description: Takes a DayNode and connects it to the end of the linked list

Parameter: DayNode newnode

Return type: void

Preconditions:

Post conditions:

Algorithm:

For DayNode temp = head; temp!= null; temp = temp.getLink()

If temp.getLink() = null

temp.setLink(newnode)

end if

if head = null

head = newnode

end if

**Sort**

Description: Sorts the nodes in the linked list according to either ascending order or descending order of due date.

Parameter: Boolean ascending

Return type: void

Preconditions: A linked list of DayNodes exists

Post conditions: The linked list is sorted

Algorithm:

Bubble sort comparing due date of adjacent nodes

The bubble sort is reversed goring from back to front when sorting in ascending order

**Main Menu Class**

Description: This class will function as the main menu of the program allowing the user to choose different operations.

Parameter:

Return type: void

Preconditions:

Post conditions:

Algorithm:

While int loop = 0

loop always = 0

Print “choices of process the user would want to use”

Get from the user the choice of process

If int choice = particular numerical choice

Carry out the task

End if

End loop

**DayManipulation Class**

**Adding information**

Description: Takes information inputted by the user and edits the appropriate Day class to reflect the new information.

Parameter: String schoolwork, String newinfo, int duemonth, int dueday, int assignedmonth, int assignedday

Return type: void

Preconditions: A spot for the new information must be available within the Day object

Post conditions:

Algorithm:

\* Example below is for the case that the month is January and the information entered was classified as homework. In instances of other months or other school work such as quiz, the algorithm will be the same but different variables will be used.

If schoolwork equals homework

If assignedmonth equals to 1

For int a = 0; January.getDayList(assignedday-1).getHomeWork(a) not equal null && a < 10; a++

int count ++;

if count does not equal to 9

January.getDayList(assignedday-1).setHomeWork(count-1, newinfo);

January.getDayList(assignedday-1).setDueDateHomeWork(count-1, duemonth, dueday);

January.getDayList(assignedday-1).setHomeWorkState(count-1, “incomplete”) ;

End if

End if

**Deleting information**

Description: Deletes information in Day objects to reflect the appropriate changes.

Parameter: String schoolwork, int index, int assignedmonth, int assignedday

Return type: void

Preconditions:

Post conditions:

Algorithm:

\* Example below is for the case that the month is January and the information entered was classified as homework. In instances of other months or other school work such as quiz, the algorithm will be the same but different variables will be used.

If schoolwork equals homework

If assignedmonth equals to 1

January.getDayList(assignedday-1).setHomeWork(index, null);

End if

End if

**Marking a piece of school work as complete**

Description: Edits the Day object to reflect that a task has been completed.

Parameter: String schoolwork, int index, int assignedmonth, int assignedday

Return type: void

Preconditions:

Post conditions:

Algorithm:

\* Example below is for the case that the month is January and the information entered was classified as homework. In instances of other months or other school work such as quiz, the algorithm will be the same but different variables will be used.

If schoolwork equals homework

If assignedmonth equals to 1

January.getDayList(assignedday-1).setHomeWorkState(count-1, “complete”) ;

End if

End if

**Displaying stored information about schoolwork assigned on a particular day**

Description: Displays stored information about a particular type of school work assigned on a particular day to the screen.

Parameter: String schoolwork, int assignedmonth, int assignedday

Return type: void

Preconditions: There should be saved data about the schoolwork

Post conditions: Information is displayed

Algorithm:

\* Example below is for the case that the month is January and the information entered was classified as homework. In instances of other months or other school work such as quiz, the algorithm will be the same but different variables will be used.

If schoolwork equals homework

If assignedmonth equals to 1

For (int a = 0; January.getDayList(assignedday-1).getHomeWork (a) != null && a < 10; a increase by 1)

January.getDayList(assignedday-1).toStringHomeWork(a)

End for loop

End if

End if

**Quick Check**

Description: searches through every Day object to see if any of a specific type of schoolwork is due within 7 days of the present date. The Day objects are put into a linked list and returned.

Parameter: String schoolwork, int assignedmonth, int assignedday

Return type: DayQueue

Preconditions: The present date must be known

Post conditions: Creates a linked list of all Day objects with schoolwork due in next 7 days

Algorithm:

\* Example below is for the case that the month is January and the information entered was classified as homework. In instances of other months or other school work such as quiz, the algorithm will be the same but different variables will be used.

If schoolwork equals homework

For (int a = 0; a < 31; a ++)

For (int b = 0; && b < 10; a ++)

If assignedday < 25

If (January.getDayList(a).getDueDateHomeWork (b, 0) = assignedmonth && January.getDayList(a).getDueDateHomeWork (b, 1) greater equal to assignedday && January.getDayList(a).getDueDateHomeWork (b, 1) smaller equal to assignedday + 7)

Append January.getDayList(a) to end of DayQueue with method appendEnd() from DayQueue class

End if

Else

If (January.getDayList(a).getDueDateHomeWork (b, 0) = assignedmonth && January.getDayList(a).getDueDateHomeWork (b, 1) greater equal to assignedday && January.getDayList(a).getDueDateHomeWork (b, 1) smaller equal to assignedday + (31 - assignedday)

Append January.getDayList(a) to end of DayQueue with method appendEnd() from DayQueue class

End if

Else if (January.getDayList(a).getDueDateHomeWork (b, 0) = assignedmonth+1 && January.getDayList(a).getDueDateHomeWork (b, 1) greater equal to 0 && January.getDayList(a).getDueDateHomeWork (b, 1) smaller equal to 7 – (31 – assignedday))

Append January.getDayList(a) to end of DayQueue with method appendEnd() from DayQueue class

End Else If

End Else

End for loop

End for loop

End if

**Due Date Search**

Description: searches through every Day object to see if any of a specific type of schoolwork is due on a specific day chosen by the user. The Day objects are put into a linked list and returned.

Parameter: String schoolwork, int duemonth, int dueday

Return type: DayQueue

Preconditions:

Post conditions: Creates a linked list of all Day objects with schoolwork due in the specified day

Algorithm:

\* Example below is for the case that the month is January and the information entered was classified as homework. In instances of other months or other school work such as quiz, the algorithm will be the same but different variables will be used.

If schoolwork equals homework

For (int a = 0; a < 31; a ++)

For (int b = 0; && b < 10; a ++)

If (January.getDayList(a).getDueDateHomeWork (b, 0) = duemonth && January.getDayList(a).getDueDateHomeWork (b, 1) = dueday)

Append January.getDayList(a) to end of DayQueue with method appendEnd() from DayQueue class

End if

End for loop

End for loop

End if

**Reading Data from Saved File**

Description: Reads the data from the saved file and saves it into Day objects when the program is first started.

Parameter: none

Return type: void

Preconditions: the saved file must exist

Post conditions: The information is transferred to memory

Algorithm:

\* Example below is for the case of the month of month is January. In instances of other months the algorithm will be the same but different variables will be used.

For (int a = 0; a < 31; a++)

For (int b = 0; b < number of lines of information for each Day class; b++)

January.set[a].set(first type of data in list of information)(read.nextline());

Repeate for all information

End for

End for

**Writing Data to Saved File**

Description: Rewrite the saved files to reflect the changes made by the user when the user decides to exit the program.

Parameter: none

Return type: void

Preconditions:

Post conditions: The saved files are updated

Algorithm:

\* Example below is for the case of the month of January. In instances of other months the algorithm will be the same but different variables will be used.

For (int a = 0; a < 31; a++)

For (int b = 0; b < number of lines of information for each Day class; b++)

Write line January.set[a].get(first type of data in list of information);

Repeate for all information

End for

End for

# B3: Modular Organization

\*All fields and methods can be found in the previous sections above

ThirtyDay Class

Day Class

ThirtyOneDay Class

TwentyEightDay Class

DayNode Class

DayQueue Class

DayManipulation Class

Main Menu Class

The diagram above shows the interactions between the 8 classes. The Day class will be stored in the TwentyEightDay, ThirtyDay, and ThirtyOneDay classes for better organization. These organized month classes will be called by the DayManipulation to directly alter the Day classes that they hold; these classes are also accessed by the DayNode Class in order to create a linked list containing the appropriate Day Classes.

--------------------------

Stage C – Program

Peng Fei Wang

----------------------------

# Stage C – Program

# C1 Using good programming style

**import** java.io.IOException;

**import** java.util.\*;

/\*\*

\* **@author** Peng Fei Wang

\* Date finished: Feburary 28th 2013

\* School: Victoria Park CI

\* Computer Used: home laptop Acer Aspire 5741-5193

\* This program is an electronic agenda that allows the user to enter school related work. The work is stored within saved files

\* and the user can review the work at a later time through searching by date assigned or due date. The user can also edit the

\* work stored at a later time.

\*

\*/

/\*\*

\* **@author** Peng Fei Wang

\*the Menu class is a menu used to allow the user to choice and navigate through the program.

\*/

**public** **class** Menu {

**static** **int** *date*;

**public** **static** **void** main (String[]args) **throws** IOException{

**try**{

Scanner sc = **new** Scanner(System.*in*);

**int** loop = 0, secondaryloop = 0;

**int** input = 0;;

Functions f1 = **new** Functions();

f1.saveFileInitiation();

*date* = f1.askDate();

**while** (loop == 0) // infinite loop of the menu, can only escape with the exit function

{

System.*out*.println("Main Menu");

System.*out*.println("-------------------------------------------------------------");

System.*out*.println("1. Add Information");

System.*out*.println("2. Retrive/Edit Previous Information");

System.*out*.println("3. Quick Check");

System.*out*.println("4. Search for SchoolWork/Activity with respect to due date");

System.*out*.println("5. Exit");

System.*out*.println("Enter your option:");

**try** {

input = sc.nextInt();

}

**catch** (InputMismatchException e)

{

sc.nextLine();

}

**if** (input == 1)

{

f1.addInformation(f1.loadDate(*date*)); // calls the addInformation method to add information

}

**else** **if** (input == 2)

{

f1.editInformation();// calls the editInformation method to edit information

}

**else** **if** (input == 3)

{

f1.quickCheck(); // calls the quickCheck method to perform a quick check

}

**else** **if** (input == 4)

{

f1.check(); // calls the check method to perform a check

}

**else** **if** (input == 5)

{

System.*exit*(0); // exists the program

}

**else** **if** ( input > 5 | input < 1)

{

System.*out*.println("Inappropriate Input");

}

**else** {

sc.nextLine();

System.*out*.println("Inappropriate Input");

}

}

}

**catch** (Exception e){

System.*out*.println("An unexpected error occured");

}

}

}

import java.io.\*;

import java.util.InputMismatchException;

import java.util.Scanner;

/\*\*

\* @author Peng Fei Wang

\* This class stores all the functions that the program requires to function. It extends to the menu class to access

\* the date field.

\*

\*/

public class Functions extends Menu {

/\*\*

\* the saveFileInitiation method is used to check if saved files with the appropriate names exist and are readable

\* if, the saved files do not exist, then this method can create new blank saved files with the appropriate names.

\* @throws IOException

\*/

public void saveFileInitiation () throws IOException

{

String namepart1;//the string that serves as the name of save files

String name;//string that represents the complete name of a save file

int choice;//int that represents user input

Scanner sc = new Scanner (System.in); // used to get user input

try {

for (int b = 1; b < 366; b++) // loop is used to go through all expected save files

{

namepart1 = Integer.toString(b);

name = namepart1 + ".txt";

File file = new File (name);

if (file.canRead() == false) // check to see if the file can be read

{

System.out.println("Saved Files are either missing or the program is denied access");

System.out.println("1. Create new SaveFiles");

System.out.println("2. Exit");

System.out.println("Choose an option: ");

choice = sc.nextInt();

if (choice == 1)

{

for (int a = 1; a < 366; a++) // creates new blank saved files

{

namepart1 = Integer.toString(a);

name = namepart1 + ".txt";

FileWriter write2 = new FileWriter (name);

BufferedWriter write = new BufferedWriter (write2);

for (int z = 0; z < 60; z++)

{

write.newLine();

}

write.close();

}

}

else if (choice == 2)

{

System.exit(0);

}

}

}

}

catch (IOException e) // catch io exceptions

{

}

//String check = read.readLine();

//System.out.println("check");

}

/\*\*

\* the loadDate class is used to read a specific save file and transfer that information into a Day object.

\* @param day: an int that represents the identity of the save file to be loaded

\* @throws IOException

\*/

public Day loadDate(int day) throws IOException

{

String name;//String that is the name of the save file that will be read

name = Integer.toString(day);

int temp2;//an int used as a temporary storage place before the information is placed into the Day object

String temp;//a string used as a temporary storage place before the information is placed into the Day object

boolean temp3;//a boolean used as a temporary storage place before the information is placed into the Day object

try {

FileReader read2 = new FileReader (name+".txt");

BufferedReader read = new BufferedReader (read2);

Day current = new Day();

for (int a = 0; a < 5; a++) //reads information and transfers the information to the Day object

{

temp = read.readLine();

current.setHomework(a, temp);

}

for (int b = 0; b < 5; b++) //reads information and transfers the information to the Day object

{

temp = read.readLine();

if (temp.equals("true"))

{

temp3 = true;

}

else temp3 = false;

current.setHomeworkstate(b, temp3);

}

for (int c = 0; c < 5; c++) //reads information and transfers the information to the Day object

{

temp = read.readLine();

try{

temp2 = Integer.parseInt(temp);

}

catch(NumberFormatException e)

{

temp2 = 0;

}

current.setHomeworkduedate(c, temp2);

}

for (int d = 0; d < 5; d++) //reads information and transfers the information to the Day object

{

temp = read.readLine();

current.setQuiz(d, temp);

}

for (int e = 0; e < 5; e++) //reads information and transfers the information to the Day object

{

temp = read.readLine();

if (temp.equals("true"))

{

temp3 = true;

}

else temp3 = false;

current.setQuizstate(e, temp3);

}

for (int f = 0; f < 5; f++) //reads information and transfers the information to the Day object

{

temp = read.readLine();

try{

temp2 = Integer.parseInt(temp);

}

catch (NumberFormatException e)//catches number format exception

{

temp2 = 0;

}

current.setQuizduedate(f, temp2);

}

for (int g = 0; g < 5; g++) //reads information and transfers the information to the Day object

{

temp = read.readLine();

current.setTest(g, temp);

}

for (int h = 0; h < 5; h++) //reads information and transfers the information to the Day object

{

temp = read.readLine();

if (temp.equals("true"))

{

temp3 = true;

}

else temp3 = false;

current.setTeststate(h, temp3);

}

for (int i = 0; i < 5; i++) //reads information and transfers the information to the Day object

{

temp = read.readLine();

try {

temp2 = Integer.parseInt(temp);

}

catch (NumberFormatException e){//catches number format exception

temp2 = 0;

}

current.setTestduedate(i, temp2);

}

for (int j = 0; j < 5; j++) //reads information and transfers the information to the Day object

{

temp = read.readLine();

current.setAssignment(j, temp);

}

for (int k = 0; k < 5; k++) //reads information and transfers the information to the Day object

{

temp = read.readLine();

if (temp.equals("true"))

{

temp3 = true;

}

else temp3 = false;

current.setAssignmentstate(k, temp3);

}

for (int l = 0; l < 5; l++) //reads information and transfers the information to the Day object

{

temp = read.readLine();

try{

temp2 = Integer.parseInt(temp);

}

catch (NumberFormatException e) //catches number format exception

{

temp2 = 0;

}

current.setAssignmentduedate(l, temp2);

}

read.close();

return current;

} catch (FileNotFoundException e) {

System.out.println("Saved file does not exist, please reload the program");

}

Day placeholder = new Day(); //place holder null object returned if an error occured in reading the saved file

return placeholder;

}

/\*\*

\* the saveDate method is used to save the information of a Day object to a save file

\* @param date: an integer that represents the name of the saved file

\* @param current: a Day object that will have its information saved to a text file.

\* @throws IOException

\*/

public void saveDate(int date, Day current) throws IOException

{

String name = Integer.toString(date);

try {

FileWriter write2 = new FileWriter (date+".txt");

BufferedWriter write = new BufferedWriter (write2);

for (int a = 0; a < 5; a++) //writes information from Day object to save file

{

write.write(current.getHomework(a));

write.newLine();

}

for (int b = 0; b < 5; b++) //writes information from Day object to save file

{

if (current.getHomeworkstate(b)==true)

{

write.write("true");

write.newLine();

}

else

{

write.write("false");

write.newLine();

}

}

for (int c = 0; c < 5; c++)//writes information from Day object to save file

{

write.write(Integer.toString(current.getHomeworkduedate(c)));

write.newLine();

}

for (int d = 0; d < 5; d++)//writes information from Day object to save file

{

write.write(current.getQuiz(d));

write.newLine();

}

for (int e = 0; e < 5; e++)//writes information from Day object to save file

{

if (current.getQuizstate(e)==true)

{

write.write("true");

write.newLine();

}

else

{

write.write("false");

write.newLine();

}

}

for (int f = 0; f < 5; f++)//writes information from Day object to save file

{

write.write(Integer.toString(current.getQuizduedate(f)));

write.newLine();

}

for (int g = 0; g < 5; g++)//writes information from Day object to save file

{

write.write(current.getTest(g));

write.newLine();

}

for (int h = 0; h < 5; h++)//writes information from Day object to save file

{

if (current.getTeststate(h)==true)

{

write.write("true");

write.newLine();

}

else

{

write.write("false");

write.newLine();

}

}

for (int i = 0; i < 5; i++)//writes information from Day object to save file

{

write.write(Integer.toString(current.getTestduedate(i)));

write.newLine();

}

for (int j = 0; j < 5; j++)//writes information from Day object to save file

{

write.write(current.getAssignment(j));

write.newLine();

}

for (int k = 0; k < 5; k++)//writes information from Day object to save file

{

if (current.getAssignmentstate(k)==true)

{

write.write("true");

write.newLine();

}

else

{

write.write("false");

write.newLine();

}

}

for (int l = 0; l < 5; l++)//writes information from Day object to save file

{

write.write(Integer.toString(current.getAssignmentduedate(l)));

write.newLine();

}

write.close();

} catch (FileNotFoundException e) // catches file not found exceptions

{

System.out.println("Saved file does not exist, please reload the program");

}

}

/\*\*the askDate method asks the user for the current date by asking for month and day. This information is converted

\* to an absolute scale where january first is represented as 1, january second is represented as 2 ... all the day to

\* december 31st.

\* @return int: the integer returned represents the absolute scale of the date inputted by the user

\*/

public int askDate()

{

Scanner sc = new Scanner(System.in);

int month = 0, day = 0, presentDate = 0;

int loop = 0;

while (loop == 0)

{

loop = 1;

System.out.println("Please Select the Current Month: ");

System.out.println("1. January ");

System.out.println("2. February ");

System.out.println("3. March ");

System.out.println("4. April ");

System.out.println("5. May ");

System.out.println("6. June ");

System.out.println("7. July ");

System.out.println("8. Auguest ");

System.out.println("9. September ");

System.out.println("10. October ");

System.out.println("11. November ");

System.out.println("12. December ");

System.out.println("Please Enter Your Choice ");

try {

month = sc.nextInt(); // asks user for month

sc.nextLine();

System.out.println("Please Enter the Present Day of the Month: ");

day = sc.nextInt(); // asks user for day

if (month == 1) // conversion from month day to absolute scale

{

presentDate = day;

}

else if (month == 2)

{

presentDate = day+31;

}

else if (month == 3)

{

presentDate = day+31+28;

}

else if (month == 4)

{

presentDate = day+31+28+31;

}

else if (month == 5)

{

presentDate = day+31+28+31+30;

}

else if (month == 6)

{

presentDate = day+31+28+31+30+31;

}

else if (month == 7)

{

presentDate = day+31+28+31+30+31+30;

}

else if (month == 8)

{

presentDate = day+31+28+31+30+31+30+31;

}

else if (month == 9)

{

presentDate = day+31+28+31+30+31+30+31+31;

}

else if (month == 10)

{

presentDate = day+31+28+31+30+31+30+31+31+30;

}

else if (month == 11)

{

presentDate = day+31+28+31+30+31+30+31+31+30+31;

}

else if (month == 12)

{

presentDate = day+31+28+31+30+31+30+31+31+30+31+30;

}

}

catch (InputMismatchException e) //catches input mistmatch exceptions

{

}

if ( month > 12 | month < 1 | day > 31 | day < 1) // catches inappropriate input

{

sc.nextLine();

System.out.println("Inappropriate Input");

loop = 0;

presentDate = 0;

}

else {

}

}

return presentDate;

}

/\*\*

\*The addInformation method takes in specific input and adds the information from the input to the day

\* @param currentDay: this it the Day object that will have additional information added to it

\* @throws IOException

\*/

public void addInformation (Day currentDay) throws IOException

{

int input, secondaryinput, newdate;

String info;

int exit = 0;

boolean error = false;

int count = 0;

boolean found = false;

Scanner sc = new Scanner (System.in);

System.out.println("Add Information:");

System.out.println("Which category does the new school work/activity fall under?");

System.out.println("1. Homework");

System.out.println("2. Assignment");

System.out.println("3. Quiz");

System.out.println("4. Test");

System.out.println("5. Return to Main Menu");

System.out.println("Please Enter Your Choice:");

try {

input = sc.nextInt();

sc.nextLine();

if (input == 1)

{

System.out.println("New Homework");

System.out.println("Is the homework due?");

System.out.println("1. Tomorrow");

System.out.println("2. The Day After Tomorrow");

System.out.println("3. Other");

secondaryinput = sc.nextInt();

sc.nextLine();

if (secondaryinput == 1)

{

for (int a = 0; a < 5; a++) //searches for a free slot out of the 5 available spaces for new homework

{

if (currentDay.getHomework(a).equals("") == true)

{

currentDay.setHomeworkduedate(a, date+1); // sets the due date

System.out.println("Please Enter Information Concerning this Homework: ");

info = sc.nextLine();

currentDay.setHomework(a, info); //sets the general information

currentDay.setHomeworkstate(a,false); //sets the state of the work

found = true;

}

if (a == 4 && found == false)

{

System.out.println("There are no more spaces for new Homework");

}

if (found == true)

{

a = 6;

}

}

}

else if (secondaryinput == 2)

{

for (int b = 0; b < 5; b++)

{

if (currentDay.getHomework(b).equals("") == true)

{

currentDay.setHomeworkduedate(b, date+2);

System.out.println("Please Enter Information Concerning this Homework: ");

info = sc.nextLine();

currentDay.setHomework(b, info);

currentDay.setHomeworkstate(b,false);

found = true;

}

count ++;

if (b == 4 && found == false)

{

System.out.println("There are no more spaces for new Homework");

}

if (found == true)

{

b = 6;

}

}

}

else if (secondaryinput == 3)

{

newdate = askDate();

for (int c = 0; c < 5; c++)

{

if (currentDay.getHomework(c).equals("") == true)

{

currentDay.setHomeworkduedate(c, newdate);

System.out.println("Please Enter Information Concerning this Homework: ");

info = sc.nextLine();

currentDay.setHomework(c, info);

currentDay.setHomeworkstate(c,false);

found = true;

}

count ++;

if (c == 5 && found == false)

{

System.out.println("There are no more spaces for new Homework");

}

if (found == true)

{

c = 6;

}

}

}

else error = true;

}

else if (input == 2)

{

System.out.println("New Assignment");

System.out.println("Is the Assignment due?");

System.out.println("1. Tomorrow");

System.out.println("2. The Day After Tomorrow");

System.out.println("3. Other");

secondaryinput = sc.nextInt();

sc.nextLine();

if (secondaryinput == 1)

{

for (int d = 0; d < 5; d++)

{

if (currentDay.getAssignment(d).equals("") == true)

{

currentDay.setAssignmentduedate(d, date+1);

System.out.println("Please Enter Information Concerning this Assignment: ");

info = sc.nextLine();

currentDay.setAssignment(d, info);

currentDay.setAssignmentstate(d,false);

found = true;

}

if (d == 4 && found == false)

{

System.out.println("There are no more spaces for new Assignment");

}

if (found == true)

{

d = 6;

}

}

}

else if (secondaryinput == 2)

{

for (int e = 0; e < 5; e++)

{

if (currentDay.getAssignment(e).equals("") == true)

{

currentDay.setHomeworkduedate(e, date+2);

System.out.println("Please Enter Information Concerning this Assignment: ");

info = sc.nextLine();

currentDay.setAssignment(e, info);

currentDay.setAssignmentstate(e,false);

found = true;

}

count ++;

if (e == 4 && found == false)

{

System.out.println("There are no more spaces for new Assignment");

}

if (found == true)

{

e = 6;

}

}

}

else if (secondaryinput == 3)

{

newdate = askDate();

for (int f = 0; f < 5; f++)

{

if (currentDay.getAssignment(f).equals("") == true)

{

currentDay.setAssignmentduedate(f, newdate);

System.out.println("Please Enter Information Concerning this Assignment: ");

info = sc.nextLine();

currentDay.setAssignment(f, info);

currentDay.setAssignmentstate(f,false);

found = true;

}

count ++;

if (f == 5 && found == false)

{

System.out.println("There are no more spaces for new Assignment");

}

if (found == true)

{

f = 6;

}

}

}

else error = true;

}

else if (input == 3)

{

System.out.println("New Quiz");

System.out.println("Is the Quiz due?");

System.out.println("1. Tomorrow");

System.out.println("2. The Day After Tomorrow");

System.out.println("3. Other");

secondaryinput = sc.nextInt();

sc.nextLine();

if (secondaryinput == 1)

{

for (int g = 0; g < 5; g++)

{

if (currentDay.getQuiz(g).equals("") == true)

{

currentDay.setQuizduedate(g, date+1);

System.out.println("Please Enter Information Concerning this Quiz: ");

info = sc.nextLine();

currentDay.setQuiz(g, info);

currentDay.setQuizstate(g,false);

found = true;

}

if (g == 4 && found == false)

{

System.out.println("There are no more spaces for new Quiz");

}

if (found == true)

{

g = 6;

}

}

}

else if (secondaryinput == 2)

{

for (int h = 0; h < 5; h++)

{

if (currentDay.getQuiz(h).equals("") == true)

{

currentDay.setQuizduedate(h, date+2);

System.out.println("Please Enter Information Concerning this Quiz: ");

info = sc.nextLine();

currentDay.setQuiz(h, info);

currentDay.setQuizstate(h,false);

found = true;

}

count ++;

if (h == 4 && found == false)

{

System.out.println("There are no more spaces for new Quiz");

}

if (found == true)

{

h = 6;

}

}

}

else if (secondaryinput == 3)

{

newdate = askDate();

for (int i = 0; i < 5; i++)

{

if (currentDay.getQuiz(i).equals("") == true)

{

currentDay.setQuizduedate(i, newdate);

System.out.println("Please Enter Information Concerning this Quiz: ");

info = sc.nextLine();

currentDay.setQuiz(i, info);

currentDay.setQuizstate(i,false);

found = true;

}

count ++;

if (i == 5 && found == false)

{

System.out.println("There are no more spaces for new Quiz");

}

if (found == true)

{

i = 6;

}

}

}

else error = true;

}

else if (input == 4)

{

System.out.println("New Test");

System.out.println("Is the Test due?");

System.out.println("1. Tomorrow");

System.out.println("2. The Day After Tomorrow");

System.out.println("3. Other");

secondaryinput = sc.nextInt();

sc.nextLine();

if (secondaryinput == 1)

{

for (int j = 0; j < 5; j++)

{

if (currentDay.getTest(j).equals("") == true)

{

currentDay.setTestduedate(j, date+1);

System.out.println("Please Enter Information Concerning this Test: ");

info = sc.nextLine();

currentDay.setTest(j, info);

currentDay.setTeststate(j,false);

found = true;

}

if (j == 4 && found == false)

{

System.out.println("There are no more spaces for new Test");

}

if (found == true)

{

j = 6;

}

}

}

else if (secondaryinput == 2)

{

for (int k = 0; k < 5; k++)

{

if (currentDay.getTest(k).equals("") == true)

{

currentDay.setTestduedate(k, date+2);

System.out.println("Please Enter Information Concerning this Test: ");

info = sc.nextLine();

currentDay.setTest(k, info);

currentDay.setTeststate(k,false);

found = true;

}

count ++;

if (k == 4 && found == false)

{

System.out.println("There are no more spaces for new Test");

}

if (found == true)

{

k = 6;

}

}

}

else if (secondaryinput == 3)

{

newdate = askDate();

for (int l = 0; l < 5; l++)

{

if (currentDay.getTest(l).equals("") == true)

{

currentDay.setTestduedate(l, newdate);

System.out.println("Please Enter Information Concerning this Test: ");

info = sc.nextLine();

currentDay.setTest(l, info);

currentDay.setTeststate(l,false);

found = true;

}

count ++;

if (l == 5 && found == false)

{

System.out.println("There are no more spaces for new Test");

}

if (found == true)

{

l = 6;

}

}

}

else error = true;

}

else if (input == 5)

{

exit = 1;

}

else

{

error = true;

}

if (error == false && exit != 1)

{

saveDate(date, currentDay); // saves the Day object to the save file

System.out.println("Action Completed");

}

else if (exit == 1)

{

}

else System.out.println("Inappropriate Input");

}

catch (InputMismatchException e)

{

System.out.println("Inappropriate input");

}

}

/\*\*

\* the editInformation method alters the information stored in a Day object to new user inputed information

\* @throws IOException

\*/

public void editInformation () throws IOException

{

int input, secondaryinput, newdate = 0, choice, worknumber;

String info;

int end = 0;

boolean error = false;

boolean found = false;

Scanner sc = new Scanner (System.in);

System.out.println("Which Day would you like to revisit?");

System.out.println("1. Today");

System.out.println("2. Yesterday");

System.out.println("3. The Day Before Yesterday");

System.out.println("4. Other");

try{

input = sc.nextInt(); // gets file name

sc.nextLine();

if (input == 1)

{

newdate = date;

}

else if (input == 2)

{

newdate = date-1;

}

else if (input == 3)

{

newdate = date -2;

}

else if (input == 4)

{

newdate = askDate();

}

else error = true;

if (error != true)

{

System.out.println("Which category does the new school work/activity fall under?");

System.out.println("1. Homework");

System.out.println("2. Assignment");

System.out.println("3. Quiz");

System.out.println("4. Test");

System.out.println("5. Return to Main Menu");

System.out.println("Please Enter Your Choice:");

input = sc.nextInt();

sc.nextLine();

Day day = new Day();

day = loadDate(newdate);

if (input == 1)

{

System.out.println("Please Choose which Homework to Edit");

for (int y = 0; y < 5; y++)

{

System.out.println(y+1+". "+day.getHomework(y));

}

System.out.println("6. Return to Main Menu");

worknumber = sc.nextInt();

sc.nextLine();

if (worknumber < 6 && worknumber > 0){

System.out.println("Please Select Option");

System.out.println("1. Check Completed");

System.out.println("2. Edit Information");

choice = sc.nextInt();

sc.nextLine();

if (choice == 1 && day.getHomework(worknumber-1).equals("") != true)

{

day.setHomeworkstate(worknumber-1, true);

}

else if (choice == 2&& day.getHomework(worknumber-1).equals("") != true)

{

System.out.println("Enter Revised Information");

info = sc.nextLine();

day.setHomework(worknumber-1, info);

}

}else System.out.println("Inappropriate Input");

}

else if (input == 2)

{

System.out.println("Please Choose which Assignment to Edit");

for (int x = 0; x < 5; x++)

{

System.out.println(x+1+". "+day.getAssignment(x));

}

System.out.println("6. Return to Main Menu");

worknumber = sc.nextInt();

sc.nextLine();

if (worknumber < 6 && worknumber > 0){

System.out.println("Please Select Option");

System.out.println("1. Check Completed");

System.out.println("2. Edit Information");

choice = sc.nextInt();

sc.nextLine();

if (choice == 1)

{

day.setAssignmentstate(worknumber-1, true);

}

else if (choice == 2)

{

System.out.println("Enter Revised Information");

info = sc.nextLine();

day.setAssignment(worknumber-1, info);

}

}

else System.out.println("Inappropriate Input");

}

else if (input == 3)

{

System.out.println("Please Choose which Quiz to Edit");

for (int v = 0; v < 5; v++)

{

System.out.println(v+1+". "+day.getQuiz(v));

}

System.out.println("6. Return to Main Menu");

worknumber = sc.nextInt();

sc.nextLine();

if (worknumber < 6 && worknumber > 0){

System.out.println("Please Select Option");

System.out.println("1. Check Completed");

System.out.println("2. Edit Information");

choice = sc.nextInt();

sc.nextLine();

if (choice == 1)

{

day.setQuizstate(worknumber-1, true);

}

else if (choice == 2)

{

System.out.println("Enter Revised Information");

info = sc.nextLine();

day.setQuiz(worknumber-1, info);

}

}

else System.out.println("Inappropriate Input");

}

else if (input == 4)

{

System.out.println("Please Choose which Test to Edit");

for (int w = 0; w < 5; w++)

{

System.out.println(w+1+". "+day.getTest(w));

}

System.out.println("6. Return to Main Menu");

worknumber = sc.nextInt();

sc.nextLine();

if (worknumber < 6 && worknumber > 0){

System.out.println("Please Select Option");

System.out.println("1. Check Completed");

System.out.println("2. Edit Information");

choice = sc.nextInt();

sc.nextLine();

if (choice == 1)

{

day.setTeststate(worknumber-1, true);

}

else if (choice == 2)

{

System.out.println("Enter Revised Information");

info = sc.nextLine();

day.setTest(worknumber-1, info);

}

}

else System.out.println("Inappropriate Input");

}

else if (input == 5)

{

end = 1;

}

else

{

error = true;

}

if (error == false && end != 1)

{

saveDate(date, day); // saves the Day object to the save file

System.out.println("Action Completed");

}

else if (end == 1)

{

}

else System.out.println("Inappropriate input");

}

else

{

System.out.println("Inappropriate input");

}

}

catch (InputMismatchException e)

{

System.out.println("Inappropriate input");

}

}

/\*\*

\* the quickCheck method searches through all the save files to check if any school work is due within 7 days of the

\* present day

\* @throws IOException

\*/

public void quickCheck () throws IOException

{

int input, secondaryinput, newdate = 0, choice, worknumber;

String info, name;

boolean found = false;

boolean error = false;

Scanner sc = new Scanner (System.in);

try{

System.out.println("Quick Check");

System.out.println("1. All homework due in the next 7 days");

System.out.println("2. All assignments due in the next 7 days");

System.out.println("3. All quizzes occurring in the next 7 days");

System.out.println("4. All tests occurring in the next 7 days");

input = sc.nextInt();

sc.nextLine();

int loop = 0, traverse = 1;

LinkedList list = new LinkedList(); // creates a linked list to contain the Day objects the meet the 7 days criteria

if (input == 1)

{

System.out.println("List of Homework due in the next 7 days: ");

for (int b = 1; b < 366; b++) // loop that searches through all save files

{

Day day = new Day ();

day = loadDate(b); //loads the Day object to check its contents

for (int a = 0; a<5; a++)

{

if (day.getHomeworkduedate(a) < date + 7 && day.getHomeworkduedate(a) >= date) //checks if due date is within 7 days

{

Node newnode = new Node (day.getHomeworkduedate(a), day); // makes a new node that contains the Day object that fit the criteria

list.appendFront(newnode); // adds the new node to the linked list

System.out.print(traverse + ". "+ day.getHomework(a)+ " Due: ");// prints the school work information

dateGenerator(day.getHomeworkduedate(a)); // converts absolute date to month, day format

if (day.getHomeworkstate(a) == true)

{

System.out.println("finished");

}

else System.out.println("unfinished");

traverse ++;

}

}

}

System.out.println("Enter an Integer to Return to Main Menu");

input = sc.nextInt();

sc.nextLine();

}

else if (input == 2)

{

System.out.println("List of Assignment due in the next 7 days: ");

for (int c = 1; c < 366; c++)

{

Day day = new Day ();

day = loadDate(c);

for (int d = 0; d<5; d++)

{

if (day.getAssignmentduedate(d) < date + 7 && day.getAssignmentduedate(d) >= date)

{

Node newnode = new Node (day.getAssignmentduedate(d), day);

list.appendFront(newnode);

System.out.print(traverse + ". "+ day.getAssignment(d)+ " Due: ");

dateGenerator(day.getAssignmentduedate(d));

if (day.getAssignmentstate(d) == true)

{

System.out.println("finished");

}

else System.out.println("unfinished");

traverse ++;

}

}

}

System.out.println("Enter an Integer to Return to Main Menu");

input = sc.nextInt();

sc.nextLine();

}

else if (input == 3)

{

System.out.println("List of Quiz due in the next 7 days: ");

for (int e = 1; e < 366; e++)

{

Day day = new Day ();

day = loadDate(e);

for (int f = 0; f<5; f++)

{

if (day.getQuizduedate(f) < date + 7 && day.getQuizduedate(f) >= date)

{

Node newnode = new Node (day.getQuizduedate(f), day);

list.appendFront(newnode);

System.out.print(traverse + ". "+ day.getQuiz(f)+ " Due: ");

dateGenerator(day.getQuizduedate(f));

if (day.getQuizstate(f) == true)

{

System.out.println("finished");

}

else System.out.println("unfinished");

traverse ++;

}

}

}

System.out.println("Enter an Integer to Return to Main Menu");

input = sc.nextInt();

sc.nextLine();

}

else if (input == 4)

{

System.out.println("List of Tests due in the next 7 days: ");

for (int g = 1; g < 366; g++)

{

Day day = new Day ();

day = loadDate(g);

for (int h = 0; h<5; h++)

{

if (day.getTestduedate(h) < date + 7 && day.getTestduedate(h) >= date)

{

Node newnode = new Node (day.getTestduedate(h), day);

list.appendFront(newnode);

System.out.print(traverse + ". "+ day.getTest(h)+ " Due: ");

dateGenerator(day.getTestduedate(h));

if (day.getTeststate(h) == true)

{

System.out.println("finished");

}

else System.out.println("unfinished");

traverse ++;

}

}

}

System.out.println("Enter an Integer to Return to Main Menu");

input = sc.nextInt();

sc.nextLine();

}

}

catch (InputMismatchException e)

{

System.out.println("Inappropriate input");

}

}

/\*\*

\* the check method finds all school work due on a specific user entered date and displays this list.

\* @throws IOException

\*/

public void check () throws IOException

{

int input, secondaryinput, newdate = 0, choice, worknumber;

String info, name;

boolean found = false;

boolean error = false;

Scanner sc = new Scanner (System.in);

System.out.println("Search for school work/activity in respect to due date:");

System.out.println("Which day is the school work/activity due?");

System.out.println("1. Tomorrow");

System.out.println("2. The Day After Tomorrow");

System.out.println("3. Other");

try{

input = sc.nextInt();

sc.nextLine();

if (input == 1)

{

newdate = date+ 1;

}

else if (input == 2)

{

newdate = date + 2;

}

else if (input == 3)

{

newdate = askDate();

}

else error = true;

if (error != true){

int loop = 0, traverse = 1;

LinkedList list = new LinkedList(); // linked list that stores the list of school work due on the specified day

System.out.println("1. Display only Homework");

System.out.println("2. Display only Assignment");

System.out.println("3. Display only Quiz");

System.out.println("4. Display only Test");

input = sc.nextInt();

sc.nextLine();

if (input == 1)

{

System.out.println("List of Homework due on specified date:");

for (int b = 1; b < 366; b++) // searches all saved files to find school work that is due on specified date

{

Day day = new Day ();

day = loadDate(b);

for (int a = 0; a<5; a++)

if (day.getHomeworkduedate(a) == newdate)

{

Node newnode = new Node (day.getHomeworkduedate(a), day); // node of Day object with school work due on the specific day

list.appendFront(newnode); // adds the node to the linked list

System.out.print(traverse + ". "+ day.getHomework(a)+" Due: "); // displays the data from the Day object

dateGenerator(day.getHomeworkduedate(a));

if (day.getHomeworkstate(a) == true)

{

System.out.println("finished");

}

else System.out.println("unfinished");

traverse ++;

}

}

System.out.println("Enter an Integer to Return to Main Menu");

input = sc.nextInt();

sc.nextLine();

}

else if (input == 2)

{

System.out.println("List of Assignment due on specified date:");

for (int c = 1; c < 366; c++)

{

Day day = new Day ();

day = loadDate(c);

for (int d = 0; d<5; d++)

if (day.getAssignmentduedate(d) == newdate)

{

Node newnode = new Node (day.getAssignmentduedate(d), day);

list.appendFront(newnode);

System.out.print(traverse + ". "+ day.getAssignment(d)+" Due: ");

dateGenerator(day.getAssignmentduedate(d));

if (day.getAssignmentstate(d) == true)

{

System.out.println("finished");

}

else System.out.println("unfinished");

traverse ++;

}

}

System.out.println("Enter an Integer to Return to Main Menu");

input = sc.nextInt();

sc.nextLine();

}

else if (input == 3)

{

System.out.println("List of Quiz due on specified date:");

for (int e = 1; e < 366; e++)

{

Day day = new Day ();

day = loadDate(e);

for (int f = 0; f<5; f++)

if (day.getQuizduedate(f) == newdate)

{

Node newnode = new Node (day.getQuizduedate(f), day);

list.appendFront(newnode);

System.out.print(traverse + ". "+ day.getQuiz(f)+" Due: ");

dateGenerator(day.getQuizduedate(f));

if (day.getQuizstate(f) == true)

{

System.out.println("finished");

}

else System.out.println("unfinished");

traverse ++;

}

}

System.out.println("Enter an Integer to Return to Main Menu");

input = sc.nextInt();

sc.nextLine();

}

else if (input == 4)

{

System.out.println("List of Test due on specified date:");

for (int g = 1; g < 366; g++)

{

Day day = new Day ();

day = loadDate(g);

for (int h = 0; h<5; h++)

if (day.getTestduedate(h) == newdate)

{

Node newnode = new Node (day.getTestduedate(h), day);

list.appendFront(newnode);

System.out.print(traverse + ". "+ day.getTest(h)+" Due: ");

dateGenerator(day.getTestduedate(h));

if (day.getTeststate(h) == true)

{

System.out.println("finished");

}

else System.out.println("unfinished");

traverse ++;

}

}

System.out.println("Enter an Integer to Return to Main Menu");

input = sc.nextInt();

sc.nextLine();

}

}

else System.out.println("Inappropriate Input");

}

catch (InputMismatchException e)

{

System.out.println("Inappropriate Input");

}

}

/\*\*

\* the dateGenerator method takes date represented as integers from 1-365 and converts it to the format month, day

\* @param date

\*/

public void dateGenerator (int date)

{

String temp;

if (date - (31+28+31+30+31+30+31+31+30+31+30) > 0)

{

temp = Integer.toString(date-(31+28+31+30+31+30+31+31+30+31+30));

System.out.println("December " + temp );

}

else if (date - (31+28+31+30+31+30+31+31+30+31) > 0)

{

temp = Integer.toString(date-(31+28+31+30+31+30+31+31+30+31));

System.out.println("November " + temp );

}

else if (date - (31+28+31+30+31+30+31+31+30) > 0)

{

temp = Integer.toString(date-(31+28+31+30+31+30+31+31+30));

System.out.println("October " + temp );

}

else if (date - (31+28+31+30+31+30+31+31) > 0)

{

temp = Integer.toString(date-(31+28+31+30+31+30+31+31));

System.out.println("September " + temp );

}

else if (date - (31+28+31+30+31+30+31) > 0)

{

temp = Integer.toString(date-(31+28+31+30+31+30+31));

System.out.println("August " + temp );

}

else if (date - (31+28+31+30+31+30) > 0)

{

temp = Integer.toString(date-(31+28+31+30+31+30));

System.out.println("July " + temp );

}

else if (date - (31+28+31+30+31) > 0)

{

temp = Integer.toString(date-(31+28+31+30+31));

System.out.println("June " + temp );

}

else if (date - (31+28+31+30) > 0)

{

temp = Integer.toString(date-(31+28+31+30));

System.out.println("May " + temp );

}

else if (date - (31+28+31) > 0)

{

temp = Integer.toString(date-(31+28+31));

System.out.println("April " + temp );

}

else if (date - (31+28) > 0)

{

temp = Integer.toString(date-(31+28));

System.out.println("March " + temp );

}

else if (date - (31) > 0)

{

temp = Integer.toString(date-(31));

System.out.println("February " + temp );

}

else if (date > 0)

{

temp = Integer.toString(date);

System.out.println("January " + temp );

}

}

}

/\*\*

\* This class is the Node used in a linked list

\* **@author** Peter

\*

\*/

**public** **class** Node {

**private** **int** duedate;

**private** Day thisday;

**private** Node next;

Node (**int** duedate, Day thisday)

{

**this**.duedate = duedate;

**this**.thisday = thisday;

**this**.next = **null**;

}

/\*\*

\* **@return** the duedate

\*/

**public** **int** getDuedate() {

**return** duedate;

}

/\*\*

\* **@param** duedate the duedate to set

\*/

**public** **void** setDuedate(**int** duedate) {

**this**.duedate = duedate;

}

/\*\*

\* **@return** the next

\*/

**public** Node getNext() {

**return** next;

}

/\*\*

\* **@param** next the next to set

\*/

**public** **void** setNext(Node next) {

**this**.next = next;

}

/\*\*

\* **@return** the thisday

\*/

**public** Day getThisday() {

**return** thisday;

}

/\*\*

\* **@param** thisday the thisday to set

\*/

**public** **void** setThisday(Day thisday) {

**this**.thisday = thisday;

}

}

/\*\*

\* This class is used to create linked lists

\* **@author** Peng Fei Wang

\*

\*/

**public** **class** LinkedList {

**private** Node head;

/\*\*

\* the method get, returns a Node from the linked list

\* **@param** index: int used to locate the Node to be returned

\* **@return** Node

\*/

**public** Node get(**int** index)

{

Node temp = head;

Node temp2;

**int** counter = 0;

**for** (temp2 = temp; counter <= index; temp = temp2.getNext()) // traveres the linked list until the index is reached

{

**if** (counter == index)

{

**return** temp;

}

counter ++;

temp2 = temp;

}

**return** **null**;

}

/\*\*

\* the method appendFront adds a Node to the front of the linked list

\* **@param** o

\*/

**public** **void** appendFront(Node o)

{

Node temp;

temp = head;

head = o;

head.setNext(temp);

}

/\*\*

\* the method size returns the size of the linked list

\* **@return** int: represents the size of the linked list

\*/

**public** **int** size()

{

Node temp = head;

Node temp2;

**int** size = 0;

**for** (temp2 = temp; temp != **null**; temp = temp2.getNext())

{

size++;

temp2 = temp;

}

**return** size;

}

}

**public** **class** Day {

String [] homework = **new** String [5];

**boolean** [] homeworkstate = **new** **boolean** [5];

**int** [] homeworkduedate = **new** **int** [5];

String [] quiz = **new** String [5];

**boolean** [] quizstate = **new** **boolean** [5];

**int** [] quizduedate = **new** **int** [5];

String [] test = **new** String [5];

**boolean** [] teststate = **new** **boolean** [5];

**int** [] testduedate = **new** **int** [5];

String [] Assignment = **new** String [5];

**boolean** [] assignmentstate = **new** **boolean** [5];

**int** [] assignmentduedate = **new** **int** [5];

/\*\*

\* **@return** the assignment

\*/

**public** String getAssignment(**int** index) {

**return** Assignment[index];

}

/\*\*

\* **@param** assignment the assignment to set

\*/

**public** **void** setAssignment(**int** index, String assignment) {

Assignment[index] = assignment;

}

/\*\*

\* **@return** the assignmentstate

\*/

**public** **boolean** getAssignmentstate(**int** index) {

**return** assignmentstate[index];

}

/\*\*

\* **@param** assignmentstate the assignmentstate to set

\*/

**public** **void** setAssignmentstate(**int** index, **boolean** assignmentstate) {

**this**.assignmentstate[index] = assignmentstate;

}

/\*\*

\* **@return** the assignmentduedate

\*/

**public** **int** getAssignmentduedate(**int** index) {

**return** assignmentduedate[index];

}

/\*\*

\* **@param** assignmentduedate the assignmentduedate to set

\*/

**public** **void** setAssignmentduedate(**int** index, **int** assignmentduedate) {

**this**.assignmentduedate[index] = assignmentduedate;

}

/\*\*

\* **@return** the homework

\*/

**public** String getHomework(**int** index) {

**return** homework [index];

}

/\*\*

\* **@param** homework the homework to set

\*/

**public** **void** setHomework(**int** index, String homework) {

**this**.homework[index] = homework;

}

/\*\*

\* **@return** the homeworkstate

\*/

**public** **boolean** getHomeworkstate(**int** index) {

**return** homeworkstate[index];

}

/\*\*

\* **@param** homeworkstate the homeworkstate to set

\*/

**public** **void** setHomeworkstate(**int** index, **boolean** homeworkstate) {

**this**.homeworkstate[index] = homeworkstate;

}

/\*\*

\* **@return** the homeworkduedate

\*/

**public** **int** getHomeworkduedate(**int** index) {

**return** homeworkduedate[index];

}

/\*\*

\* **@param** homeworkduedate the homeworkduedate to set

\*/

**public** **void** setHomeworkduedate(**int** index, **int** homeworkduedate) {

**this**.homeworkduedate[index] = homeworkduedate;

}

/\*\*

\* **@return** the quiz

\*/

**public** String getQuiz(**int** index) {

**return** quiz[index];

}

/\*\*

\* **@param** quiz the quiz to set

\*/

**public** **void** setQuiz(**int** index, String quiz) {

**this**.quiz[index] = quiz;

}

/\*\*

\* **@return** the quizstate

\*/

**public** **boolean** getQuizstate(**int** index) {

**return** quizstate[index];

}

/\*\*

\* **@param** quizstate the quizstate to set

\*/

**public** **void** setQuizstate(**int** index, **boolean** quizstate) {

**this**.quizstate[index] = quizstate;

}

/\*\*

\* **@return** the quizduedate

\*/

**public** **int** getQuizduedate(**int** index) {

**return** quizduedate[index];

}

/\*\*

\* **@param** quizduedate the quizduedate to set

\*/

**public** **void** setQuizduedate(**int** index, **int** quizduedate) {

**this**.quizduedate[index] = quizduedate;

}

/\*\*

\* **@return** the test

\*/

**public** String getTest(**int** index) {

**return** test[index];

}

/\*\*

\* **@param** test the test to set

\*/

**public** **void** setTest(**int** index, String test) {

**this**.test[index] = test;

}

/\*\*

\* **@return** the teststate

\*/

**public** **boolean** getTeststate(**int** index) {

**return** teststate[index];

}

/\*\*

\* **@param** teststate the teststate to set

\*/

**public** **void** setTeststate(**int** index, **boolean** teststate) {

**this**.teststate[index] = teststate;

}

/\*\*

\* **@return** the testduedate

\*/

**public** **int** getTestduedate(**int** index) {

**return** testduedate[index];

}

/\*\*

\* **@param** testduedate the testduedate to set

\*/

**public** **void** setTestduedate(**int** index, **int** testduedate) {

**this**.testduedate[index] = testduedate;

}

}

# C2 Handling Errors

|  |  |  |  |
| --- | --- | --- | --- |
| # | Error | Sample Potential Error | Error Handling |
| 1 | Input Mismatch Exception | input = sc.nextInt();  Since sc.nextInt() only accepts integer values the user might accidentally enter a non-integer value such as a letter. | **try** {  input = sc.nextInt();  }  **catch** (InputMismatchException e)  {  sc.nextLine();  }  Try and Catch is used to prevent the program from crashing if such an error occurs. |
| 2 | IOException | FileWriter write2 = **new** FileWriter (name);  BufferedWriter write = **new** BufferedWriter (write2);  The program requires reading data from a saved file, if the saved file does not exist or if the program does not have permission to access that file an IOException will occur. | **try** { FileWriter write2 = **new** FileWriter (name);  BufferedWriter write = **new** BufferedWriter (write2);}  **catch** (IOException e)  {    }  Try and Catch is used to prevent the program from crashing if such an error occurs. |
| 3 | Unexpected Errors | Unexpected errors can occur | Two different precautions were implemented. 1. saveDate(*date*, currentDay);  The save file is updated every time a change is made rather than when the program exists. So an unexpected error that causes the program to become unpredictable will not result in a lose of data.  2. try{  }  **catch** (Exception e){  System.*out*.println("An unexpected error occured");  }  This try catch combination will catch all errors and prevent the program from crashing even if something unexpected occurs. |

# C3 Success of the Program

The success of the program will be evaluated based on the goals set out in A2. The set of goals and objectives from A2 are:

1. User can input as much information as needed for each school work/activity
2. Information can be deleted
3. Information can be viewed at a later time
4. Information can be sorted relative to both date assigned and date due.
5. Information can be searched relative to due date
6. Homework and assignments can be marked as complete
7. List of school work/activities due in the next 7 days can be generated
8. Have indicators such as “success” when an action is performed
9. When inappropriate data is inputted the program will re-prompt the user for appropriate data.
10. Inappropriate input by user will not crash the program
11. Actions aside from exit will always loop back to the main menu.
12. This goal is reached because information is stored as a String which can have (memory allowing) a size of 231-1. More than enough space for any useful information.
13. The information can be edited to become blank achieving the effect of deletion as it allows the program to write over it later if space runs out.
14. The edit, quick check, and check function allows the user to view information stored at a later time.
15. This goal was not achieved because it is found to be redundant because the user would only care about the due date, the date assigned is not important.
16. The check function finds the information relative to due date.
17. The user can mark homework as complete with the edit function.
18. The quick check function generates a list of school work due in the next 7 days
19. The program confirms adding information and editing information
20. Re-prompts are used in some functions such as getDate();
21. Try catch are implemented to prevent crashing
22. The main menu is in an infinite loop.

--------------------------

Stage D – Documentation

Peng Fei Wang

----------------------------

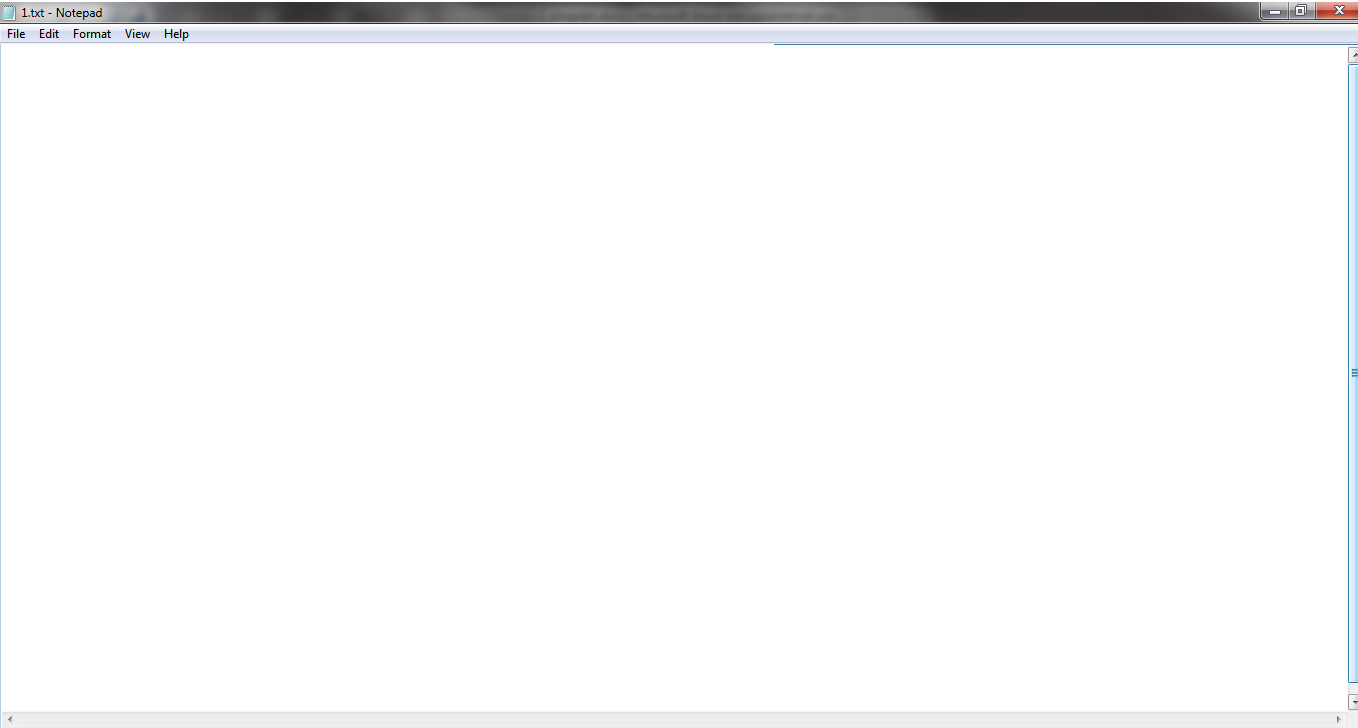
## Stage D - Documentation

# D1 Including an annotated hard copy of the test output

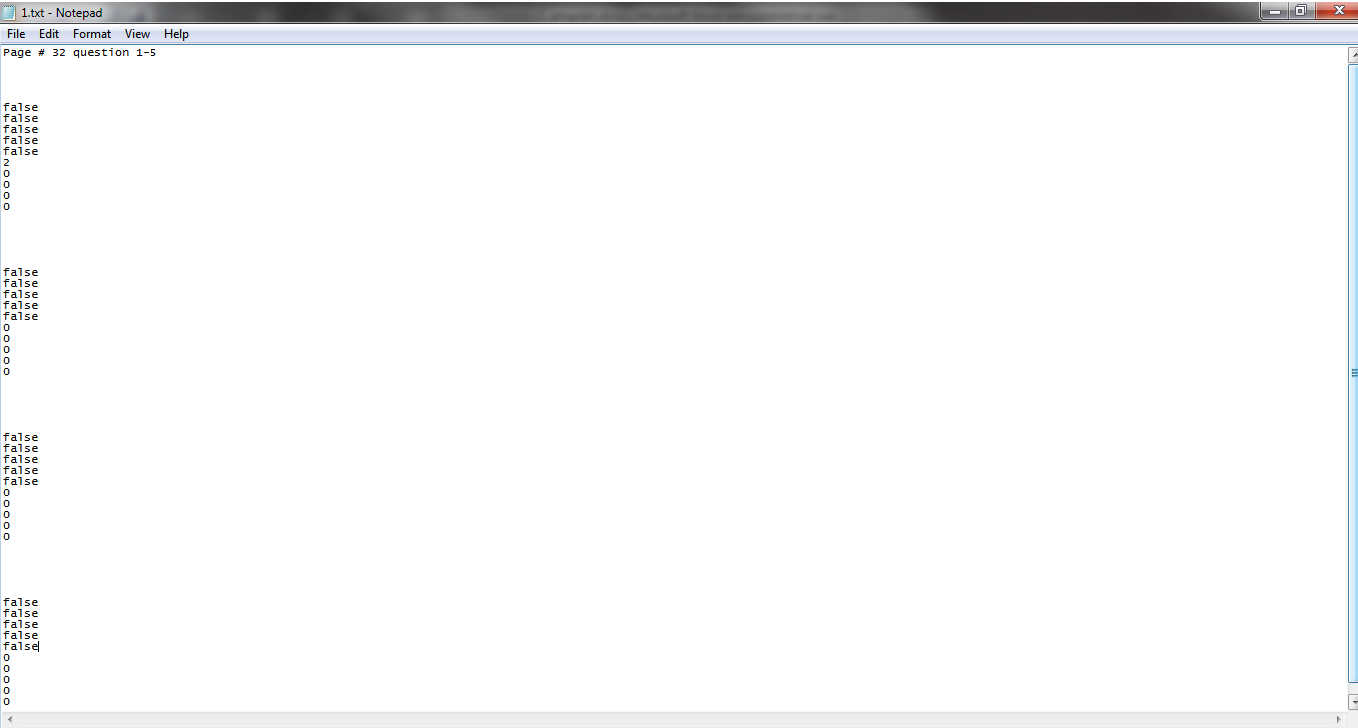
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Functionality** | **Purpose** | **Input** | **Expected Output** | **Output** |
| Loading Date | Getting the Current Date from the User (With appropriate input) | 1  1 | Please Select the Current Month:  1. January  2. February  3. March  4. April  5. May  6. June  7. July  8. Auguest  9. September  10. October  11. November  12. December  Please Enter Your Choice  1  Please Enter the Present Day of the Month:  1 | Please Select the Current Month:  1. January  2. February  3. March  4. April  5. May  6. June  7. July  8. Auguest  9. September  10. October  11. November  12. December  Please Enter Your Choice  1  Please Enter the Present Day of the Month:  1 |
|  |  |  |  |  |
| Loading Date | Getting the Current Date from the User (With inappropriate input) | -1  hi  24ewd  1  1 | Please Select the Current Month:  1. January  2. February  3. March  4. April  5. May  6. June  7. July  8. Auguest  9. September  10. October  11. November  12. December  Please Enter Your Choice  -1  Please Enter the Present Day of the Month:  hi  Inappropriate Input  Please Select the Current Month:  1. January  2. February  3. March  4. April  5. May  6. June  7. July  8. Auguest  9. September  10. October  11. November  12. December  Please Enter Your Choice  24ewd  Inappropriate Input  Please Select the Current Month:  1. January  2. February  3. March  4. April  5. May  6. June  7. July  8. Auguest  9. September  10. October  11. November  12. December  Please Enter Your Choice  1  Please Enter the Present Day of the Month:  1 | Please Select the Current Month:  1. January  2. February  3. March  4. April  5. May  6. June  7. July  8. Auguest  9. September  10. October  11. November  12. December  Please Enter Your Choice  -1  Please Enter the Present Day of the Month:  hi  Inappropriate Input  Please Select the Current Month:  1. January  2. February  3. March  4. April  5. May  6. June  7. July  8. Auguest  9. September  10. October  11. November  12. December  Please Enter Your Choice  24ewd  Inappropriate Input  Please Select the Current Month:  1. January  2. February  3. March  4. April  5. May  6. June  7. July  8. Auguest  9. September  10. October  11. November  12. December  Please Enter Your Choice  1  Please Enter the Present Day of the Month:  1 |
| Add Information | User can input as much information as needed for each schoolwork/activity | 1  1  1  Page # 32 questions 1-4 | 1. Add Information  2. Retrive/Edit Previous Information  3. Quick Check  4. Search for SchoolWork/Activity with respect to due date  5. Exit  Enter your option:  1  Add Information:  Which category does the new school work/activity fall under?  1. Homework  2. Assignment  3. Quiz  4. Test  5. Return to Main Menu  Please Enter Your Choice:  1  New Homework  Is the homework due?  1. Tomorrow  2. The Day After Tomorrow  3. Other  1  Please Enter Information Concerning this Homework:  Page # 32 questions 1-4  Action Completed | 1. Add Information  2. Retrive/Edit Previous Information  3. Quick Check  4. Search for SchoolWork/Activity with respect to due date  5. Exit  Enter your option:  1  Add Information:  Which category does the new school work/activity fall under?  1. Homework  2. Assignment  3. Quiz  4. Test  5. Return to Main Menu  Please Enter Your Choice:  1  New Homework  Is the homework due?  1. Tomorrow  2. The Day After Tomorrow  3. Other  1  Please Enter Information Concerning this Homework:  Page # 32 questions 1-4  Action Completed |
| Add information | Adding Information to homework (inappropriate input When inappropriate data is inputted the program will re-prompt the user for appropriate data)  (Inappropriate input by user will not crash the program) | -1  1  1  efd | Add Information:  Which category does the new school work/activity fall under?  1. Homework  2. Assignment  3. Quiz  4. Test  5. Return to Main Menu  Please Enter Your Choice:  -1  Inappropriate Input  Main Menu  -------------------------------------------------------------  1. Add Information  2. Retrive/Edit Previous Information  3. Quick Check  4. Search for SchoolWork/Activity with respect to due date  5. Exit  Enter your option:  1  Add Information:  Which category does the new school work/activity fall under?  1. Homework  2. Assignment  3. Quiz  4. Test  5. Return to Main Menu  Please Enter Your Choice:  1  New Homework  Is the homework due?  1. Tomorrow  2. The Day After Tomorrow  3. Other  efd  Inappropriate input | Add Information:  Which category does the new school work/activity fall under?  1. Homework  2. Assignment  3. Quiz  4. Test  5. Return to Main Menu  Please Enter Your Choice:  -1  Inappropriate Input  Main Menu  -------------------------------------------------------------  1. Add Information  2. Retrive/Edit Previous Information  3. Quick Check  4. Search for SchoolWork/Activity with respect to due date  5. Exit  Enter your option:  1  Add Information:  Which category does the new school work/activity fall under?  1. Homework  2. Assignment  3. Quiz  4. Test  5. Return to Main Menu  Please Enter Your Choice:  1  New Homework  Is the homework due?  1. Tomorrow  2. The Day After Tomorrow  3. Other  efd  Inappropriate input |
| Retrieve/edit Previous Data | Edititng previously written information (Have indicators such as “success” when an action is performed)  (Homework and assignments can be marked as complete)  (Information can be viewed at a later time) | 2  1  1  1 | 1. Add Information  2. Retrive/Edit Previous Information  3. Quick Check  4. Search for SchoolWork/Activity with respect to due date  5. Exit  Enter your option:  1  Add Information:  Which category does the new school work/activity fall under?  1. Homework  2. Assignment  3. Quiz  4. Test  5. Return to Main Menu  Please Enter Your Choice:  1  New Homework  Is the homework due?  1. Tomorrow  2. The Day After Tomorrow  3. Other  1  Please Enter Information Concerning this Homework:  Page # 32 questions 1-4  Action Completed  Main Menu  -------------------------------------------------------------  1. Add Information  2. Retrive/Edit Previous Information  3. Quick Check  4. Search for SchoolWork/Activity with respect to due date  5. Exit  Enter your option:  2  Which Day would you like to revisit?  1. Today  2. Yesterday  3. The Day Before Yesterday  4. Other  1  Which category does the new school work/activity fall under?  1. Homework  2. Assignment  3. Quiz  4. Test  5. Return to Main Menu  Please Enter Your Choice:  1  Please Choose which Homework to Edit  1. Page # 32 questions 1-4  2.  3.  4.  5.  6. Return to Main Menu  1  Please Select Option  1. Check Completed  2. Edit Information  2  Enter Revised Information  Page # 32 question 1-5  Action Completed | 1. Add Information  2. Retrive/Edit Previous Information  3. Quick Check  4. Search for SchoolWork/Activity with respect to due date  5. Exit  Enter your option:  1  Add Information:  Which category does the new school work/activity fall under?  1. Homework  2. Assignment  3. Quiz  4. Test  5. Return to Main Menu  Please Enter Your Choice:  1  New Homework  Is the homework due?  1. Tomorrow  2. The Day After Tomorrow  3. Other  1  Please Enter Information Concerning this Homework:  Page # 32 questions 1-4  Action Completed  Main Menu  -------------------------------------------------------------  1. Add Information  2. Retrive/Edit Previous Information  3. Quick Check  4. Search for SchoolWork/Activity with respect to due date  5. Exit  Enter your option:  2  Which Day would you like to revisit?  1. Today  2. Yesterday  3. The Day Before Yesterday  4. Other  1  Which category does the new school work/activity fall under?  1. Homework  2. Assignment  3. Quiz  4. Test  5. Return to Main Menu  Please Enter Your Choice:  1  Please Choose which Homework to Edit  1. Page # 32 questions 1-4  2.  3.  4.  5.  6. Return to Main Menu  1  Please Select Option  1. Check Completed  2. Edit Information  2  Enter Revised Information  Page # 32 question 1-5  Action Completed |
| Check for Homework due in next 7 days | Check for Homework due in next 7 days  (List of school work/activities due in the next 7 days can be generated) | 3  1 | 1. Add Information  2. Retrive/Edit Previous Information  3. Quick Check  4. Search for SchoolWork/Activity with respect to due date  5. Exit  Enter your option:  3  Quick Check  1. All homework due in the next 7 days  2. All assignments due in the next 7 days  3. All quizzes occurring in the next 7 days  4. All tests occurring in the next 7 days  1  List of Homework due in the next 7 days:  1. Page # 32 question 1-5 Due: January 2  unfinished  Enter an Integer to Return to Main Menu | 1. Add Information  2. Retrive/Edit Previous Information  3. Quick Check  4. Search for SchoolWork/Activity with respect to due date  5. Exit  Enter your option:  3  Quick Check  1. All homework due in the next 7 days  2. All assignments due in the next 7 days  3. All quizzes occurring in the next 7 days  4. All tests occurring in the next 7 days  1  List of Homework due in the next 7 days:  1. Page # 32 question 1-5 Due: January 2  unfinished  Enter an Integer to Return to Main Menu |
| Search for SchoolWork/Activity with respect to due date | Search for SchoolWork/Activity with respect to due date  (Information can be searched relative to due date) | 4  1  1 | 1. Add Information  2. Retrive/Edit Previous Information  3. Quick Check  4. Search for SchoolWork/Activity with respect to due date  5. Exit  Enter your option:  4  Search for school work/activity in respect to due date:  Which day is the school work/activity due?  1. Tomorrow  2. The Day After Tomorrow  3. Other  1  1. Display only Homework  2. Display only Assignment  3. Display only Quiz  4. Display only Test  1  List of Homework due on specified date:  1. Page # 32 question 1-5 Due: January 2  unfinished  Enter an Integer to Return to Main Menu | 1. Add Information  2. Retrive/Edit Previous Information  3. Quick Check  4. Search for SchoolWork/Activity with respect to due date  5. Exit  Enter your option:  4  Search for school work/activity in respect to due date:  Which day is the school work/activity due?  1. Tomorrow  2. The Day After Tomorrow  3. Other  1  1. Display only Homework  2. Display only Assignment  3. Display only Quiz  4. Display only Test  1  List of Homework due on specified date:  1. Page # 32 question 1-5 Due: January 2  unfinished  Enter an Integer to Return to Main Menu |

Proof of IO:

Saved File Before:



Saved File After:



# D2 Evaluating Solutions:

Overview:

The solution does solve the problems presented initially. By giving the user the ability to store information to a text file, the user is able to enter a string the size of 231 or until virtual memory is overloaded. Regardless that should be more than enough for practical uses. The information stored can be added and edited at another time. In addition the search function allows the user to quickly identify school work due in the near future. Indicators at the end of the program tell the user that a process is complete, and finally errors are taken account to prevent crashing of the program and in some cases looped to allow re-entry.

Limitations and possible improvements:

At this moment the agenda does not carry other important information such as which day a particular date is (ie Monday, Tuesday, etc), it does not provide information concerning holidays, late starts, and it does not tell students which day (ie Day 1, Day 2, Day 3, Day4 (different class schedule for different day)). The hard cap of 5 homework/assignment/test/quiz a day should be augmented to be a dynamic list to make information storage more efficient (there would be no blank lines in the save files) and give users more flexibility. The search function should have an option to search for all school work rather than just the specific categories. The largest improvement would be a visual representation of a calendar because the present solution requires the user to know the precise present date. With a visual calendar displayed, the user would be able to better locate the present date. The introduction of division by subject could also be implemented as different students order their agenda differently and this would give students more flexibility. The program itself should also have a function that clears the saved files in the event of the student going to a new school year.

Changes made to the Initial Solution:

Initially it was planned to have the days stored in separate month files to cut down on the number of save files and not have to convert from month, day to an absolute number representing the day. However this was more trouble than the benefits because of the quick check method. By having to search seven continuous days it is possible that the program would have to access two different month saved files. By changing to just a saved file per day, only one data file would have to be stored in the memory as opposed to over 60 (2 months). This will allow the program to operate faster and allow more strings to be entered into a particular day before the program runs out of virtual memory. The assigned date variable was removed because the name of the saved file corresponds to the date assigned, thus making it redundant. The save function was also changed so that the programmed saved after every change (adding information and edit). This minimizes the loss of data as opposed to the method outline previously which was saving only when the program exits.

Effectiveness:

The program accomplishes most of the goals and objectives set out from A2. User can input as much information as needed for each school work/activity, information can be deleted, information can be viewed at a later time. Information can be searched relative to due date, homework and assignments can be marked as complete, list of school work/activities due in the next 7 days can be generated, have indicators such as “success” when an action is performed, when inappropriate data is inputted the program will re-prompt the user for appropriate data. Inappropriate input by user will not crash the program; actions aside from exit will always loop back to the main menu. The size of storage and the search function gives this solution a distinct advantage over the traditional paper counterpart.