import java.io.\*;

import java.time.\*;

import java.time.format.\*;

import java.util.\*;

public class Assignments {

private static final File Assignments = new File("Assignments.txt");

private static final File AssignmentsTemporary = new File("AssignmentsTemp.txt");

private static final File AssignmentTimes = new File("AssignmentTimes.txt");

private static final File AssignmentTimesTemporary = new File("AssignmentTimesTemp.txt");

private static final ArrayList<String> assignmentSubjects = new ArrayList<>();

private static final ArrayList<String> assignmentNames = new ArrayList<>();

private static final ArrayList<String> assignmentDueDates = new ArrayList<>();

private static final ArrayList<String> assignmentTimesSubject = new ArrayList<>();

private static final ArrayList<String> assignmentTimesName = new ArrayList<>();

private static final ArrayList<String> assignmentTimesHours = new ArrayList<>();

private static final ArrayList<String> assignmentTimesDueDates = new ArrayList<>();

//creates a new assignment from inputs put in, writes in text files

public Assignments(String subject, String assignmentName, LocalDate formattedAssignmentDueDate, double assignmentHours,

int assignmentDays, LocalDate formattedAssignmentCompletionDate) {

try (FileWriter fw = new FileWriter("Assignments.txt", true);

BufferedWriter bw = new BufferedWriter(fw);

PrintWriter pw = new PrintWriter(bw)) {

//writes in the subject, name, and due date of the assignment into the Assignment file

pw.println(subject + "\n" + assignmentName + "\n" + formattedAssignmentDueDate);

} catch (Exception e) {

System.out.println("An error occurred while writing to the Assignments file.");

}

//calculates how many hours an assignment has to be worked on everyday

double assignmentHoursPerDay = assignmentHours / assignmentDays;

//date variable that contains the value of the first day the assignment has to be worked on

LocalDate firstAssignmentDayIteration = formattedAssignmentCompletionDate.minusDays(assignmentDays - 1);

try (FileWriter fw = new FileWriter("AssignmentTimes.txt", true);

BufferedWriter bw = new BufferedWriter(fw);

PrintWriter pw = new PrintWriter(bw)) {

//while loop that writes the name, subject, hours, and date of the assignment that has to be worked on into

//the Assignment Times file, and writes until the firstAssignmentDayIteration variable is equal to the completion day

while (firstAssignmentDayIteration.compareTo(formattedAssignmentCompletionDate) < 1) {

pw.println(assignmentName + "\n" + subject + "\n" + assignmentHoursPerDay + "\n" + firstAssignmentDayIteration);

firstAssignmentDayIteration = firstAssignmentDayIteration.plusDays(1);

}

} catch (Exception e) {

System.out.println("An error occurred while writing to the Assignment Times file.");

}

//prints the visual calendar with the new assignment printed in

Calendars.PrintCalendar();

}

public static void AssignmentPastDueDate() throws IOException {

Scanner input = new Scanner(System.in);

System.out.println("\nWhile you were gone, the following assignments were due:");

for (int i = 0; i < *assignmentDueDates*.size(); i++) {

if (LocalDate.*parse*(*assignmentDueDates*.get(i)).compareTo(Calendars.*GetCurrentDate*()) < 0) {

System.*out*.println(*assignmentSubjects*.get(i) + ": " + *assignmentNames*.get(i) + " was due on " + *assignmentDueDates*.get(i));

}

}

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

for (int i = 0; i < *assignmentDueDates*.size(); i++) {

if (LocalDate.*parse*(*assignmentDueDates*.get(i)).compareTo(Calendars.*GetCurrentDate*()) < 0) {

System.*out*.println(*assignmentSubjects*.get(i) + ": " + *assignmentNames*.get(i) + " was due on " + *assignmentDueDates*.get(i));

System.*out*.println("Did you finish this assignment? (Enter yes or no)");

String finishedAssignment = input.nextLine();

while (true) {

if (finishedAssignment.equalsIgnoreCase("Yes")) {

System.out.println("Good job! You get 1 game point, and the assignment will now be deleted.");

Games.AddGamePoints();

break;

} else if (finishedAssignment.equalsIgnoreCase("No")) {

System.out.println("Aw... that's too bad. Now deleting the assignment.");

break;

} else {

System.out.println("Please enter yes or no.");

}

}

String deleteAssignment = assignmentNames.get(i);

try {

if (AssignmentsTemporary.createNewFile()) {

System.out.println("New temporary assignment file created.");

}

} catch (Exception e) {

System.out.println("An error occurred while creating a temporary assignment file.");

}

//array with exact values from assignment file

Scanner assignmentsFileReader = new Scanner(new FileInputStream("Assignments.txt"));

ArrayList<String> assignmentFileArray = new ArrayList<>();

while (assignmentsFileReader.hasNext()) {

assignmentFileArray.add(assignmentsFileReader.nextLine());

}

assignmentsFileReader.close();

//finds the index value of the line with the same name as the assignment that needs to be deleted

int indexOfDeletedAssignment = 0;

for (int j = 0; j < assignmentFileArray.size(); j++) {

if (assignmentFileArray.get(j).equals(deleteAssignment)) {

indexOfDeletedAssignment = j + 1;

}

}

//copies assignment file to temp assignment file minus the details for the deleted assignment

//if the line in the assignment file matches the lines which we have found match the assignment we

// want to delete, then don't copy it into the new file

int assignmentFileLineNumber = 1;

Scanner assignmentsFileReader2 = new Scanner(new FileInputStream("Assignments.txt"));

BufferedWriter assignmentsTempBw = new BufferedWriter(new FileWriter(AssignmentsTemporary));

PrintWriter assignmentsTempPw = new PrintWriter(assignmentsTempBw);

while (assignmentsFileReader2.hasNext()) {

if (assignmentFileLineNumber != indexOfDeletedAssignment - 1 &&

assignmentFileLineNumber != indexOfDeletedAssignment &&

assignmentFileLineNumber != indexOfDeletedAssignment + 1) {

assignmentsTempPw.println(assignmentsFileReader2.nextLine());

} else {

assignmentsFileReader2.nextLine();

}

assignmentFileLineNumber++;

}

assignmentsTempBw.close();

assignmentsTempPw.close();

assignmentsFileReader2.close();

//deletes original assignment file with assignment we want to delete

if (!Assignments.delete()) {

System.out.println("Could not delete original assignment file.");

}

//renames the file with the deleted file removed to the original file

if (!AssignmentsTemporary.renameTo(Assignments)) {

System.out.println("Could not rename temporary file.");

}

//creates temporary assignment times file

try {

if (AssignmentTimesTemporary.createNewFile()) {

System.out.println("New temporary assignment times file created.");

}

} catch (Exception e) {

System.out.println("An error occurred while creating the temporary assignment file.");

}

//reads the subject portion of the file, and if it is equal to the name of the assignment needed

//to be deleted, skip the next 4 lines as those all pertain to the same assignment

Scanner assignmentTimesFileReader = new Scanner(new FileInputStream("AssignmentTimes.txt"));

BufferedWriter assignmentTimesTempBw = new BufferedWriter(new FileWriter(AssignmentTimesTemporary));

PrintWriter assignmentTimesTempPw = new PrintWriter(assignmentTimesTempBw);

while (assignmentTimesFileReader.hasNext()) {

String nextAssignmentTimesFileLine = assignmentTimesFileReader.nextLine();

if (nextAssignmentTimesFileLine.equals(deleteAssignment)) {

System.out.println(nextAssignmentTimesFileLine);

System.out.println(assignmentTimesFileReader.nextLine());

System.out.println(assignmentTimesFileReader.nextLine());

System.out.println(assignmentTimesFileReader.nextLine());

} else {

assignmentTimesTempPw.println(nextAssignmentTimesFileLine);

assignmentTimesTempPw.println(assignmentTimesFileReader.nextLine());

assignmentTimesTempPw.println(assignmentTimesFileReader.nextLine());

assignmentTimesTempPw.println(assignmentTimesFileReader.nextLine());

}

}

assignmentTimesTempBw.close();

assignmentTimesTempPw.close();

assignmentTimesFileReader.close();

//deletes original assignment times file with assignments we want to delete

if (!AssignmentTimes.delete()) {

System.out.println("Could not delete original assignment file.");

}

//renames the temporary assignment times file without deleted assignments to original file

if (!AssignmentTimesTemporary.renameTo(AssignmentTimes)) {

System.out.println("Could not rename temporary file.");

}

System.out.println("Deleted assignment.\n");

}

}

}

public static void DeleteAssignmentsFiles() {

if (!Assignments.delete()) {

System.out.println("Could not delete Assignments file. Must manually delete.");

} else {

System.out.println("Deleted Assignments file.");

}

if (!AssignmentTimes.delete()) {

System.out.println("Could not delete Assignments Times file. Must manually delete.");

} else {

System.out.println("Deleted Assignments Times file.");

}

}

//confirms new assignment has been created by displaying last assignment created in text files, verifying it has

// been created

public void AssignmentConfirmation() {

ReadAssignmentsFile();

int lastArrayValue = assignmentNames.size() - 1;

System.out.println("New assignment/reminder with these details successfully created:");

System.out.println(assignmentSubjects.get(lastArrayValue) + ": " + assignmentNames.get(lastArrayValue) + " " +

assignmentDueDates.get(lastArrayValue));

}

public static void AssignmentCompleteMethod() {

Scanner input = new Scanner(System.in);

ReadAssignmentTimesFile();

if (assignmentNames.isEmpty()) {

System.out.println("There are no assignments created yet, so none can be marked as done.");

return;

}

System.out.println("Here are your current assignments:");

for (int i = 0; i < assignmentDueDates.size(); i++) {

System.out.println("Subject: '" + assignmentSubjects.get(i) + "' Name: '" + assignmentNames.get(i) +

"' Due Date: '" + assignmentDueDates.get(i) + "'");

}

System.out.println("Please enter the name of the assignment you have completed:");

String deleteAssignment;

boolean deleteAssignmentError = true;

do {

try {

boolean assignmentExists = false;

while (true) {

deleteAssignment = input.nextLine();

for (String assignmentName : assignmentNames) {

if (deleteAssignment.equals(assignmentName)) {

assignmentExists = true;

break;

}

}

if (deleteAssignment.equals("-1")) {

return;

} else if (!assignmentExists) {

System.out.println("Please enter an assignment that exists.");

} else {

break;

}

}

System.out.println("Great job for completing the assignment! You get 1 game point!");

Games.AddGamePoints();

System.out.println("Now deleting assignment from text files...");

//creates temporary assignment file

try {

if (AssignmentsTemporary.createNewFile()) {

System.out.println("New temporary assignment file created.");

}

} catch (Exception e) {

System.out.println("An error occurred while creating a temporary assignment file.");

}

//array with exact values from assignment file

Scanner assignmentsFileReader = new Scanner(new FileInputStream("Assignments.txt"));

ArrayList<String> assignmentFileArray = new ArrayList<>();

while (assignmentsFileReader.hasNext()) {

assignmentFileArray.add(assignmentsFileReader.nextLine());

}

assignmentsFileReader.close();

//finds the index value of the line with the same name as the assignment that needs to be deleted

int indexOfDeletedAssignment = 0;

for (int i = 0; i < assignmentFileArray.size(); i++) {

if (assignmentFileArray.get(i).equals(deleteAssignment)) {

indexOfDeletedAssignment = i + 1;

}

}

//copies assignment file to temp assignment file minus the details for the deleted assignment

//if the line in the assignment file matches the lines which we have found match the assignment we

// want to delete, then don't copy it into the new file

int assignmentFileLineNumber = 1;

Scanner assignmentsFileReader2 = new Scanner(new FileInputStream("Assignments.txt"));

BufferedWriter assignmentsTempBw = new BufferedWriter(new FileWriter(AssignmentsTemporary));

PrintWriter assignmentsTempPw = new PrintWriter(assignmentsTempBw);

while (assignmentsFileReader2.hasNext()) {

if (assignmentFileLineNumber != indexOfDeletedAssignment - 1 &&

assignmentFileLineNumber != indexOfDeletedAssignment &&

assignmentFileLineNumber != indexOfDeletedAssignment + 1) {

assignmentsTempPw.println(assignmentsFileReader2.nextLine());

} else {

assignmentsFileReader2.nextLine();

}

assignmentFileLineNumber++;

}

assignmentsTempBw.close();

assignmentsTempPw.close();

assignmentsFileReader2.close();

//deletes original assignment file with assignment we want to delete

if (!Assignments.delete()) {

System.out.println("Could not delete original assignment file.");

}

//renames the file with the deleted file removed to the original file

if (!AssignmentsTemporary.renameTo(Assignments)) {

System.out.println("Could not rename temporary file.");

}

//creates temporary assignment times file

try {

if (AssignmentTimesTemporary.createNewFile()) {

System.out.println("New temporary assignment times file created.");

}

} catch (Exception e) {

System.out.println("An error occurred while creating the temporary assignment file.");

}

//reads the subject portion of the file, and if it is equal to the name of the assignment needed

//to be deleted, skip the next 4 lines as those all pertain to the same assignment

Scanner assignmentTimesFileReader = new Scanner(new FileInputStream("AssignmentTimes.txt"));

BufferedWriter assignmentTimesTempBw = new BufferedWriter(new FileWriter(AssignmentTimesTemporary));

PrintWriter assignmentTimesTempPw = new PrintWriter(assignmentTimesTempBw);

while (assignmentTimesFileReader.hasNext()) {

String nextAssignmentTimesFileLine = assignmentTimesFileReader.nextLine();

if (nextAssignmentTimesFileLine.equals(deleteAssignment)) {

assignmentTimesFileReader.nextLine();

assignmentTimesFileReader.nextLine();

assignmentTimesFileReader.nextLine();

} else {

assignmentTimesTempPw.println(nextAssignmentTimesFileLine);

assignmentTimesTempPw.println(assignmentTimesFileReader.nextLine());

assignmentTimesTempPw.println(assignmentTimesFileReader.nextLine());

assignmentTimesTempPw.println(assignmentTimesFileReader.nextLine());

}

}

assignmentTimesTempBw.close();

assignmentTimesTempPw.close();

assignmentTimesFileReader.close();

//deletes original assignment times file with assignments we want to delete

if (!AssignmentTimes.delete()) {

System.out.println("Could not delete original assignment file.");

}

//renames the temporary assignment times file without deleted assignments to original file

if (!AssignmentTimesTemporary.renameTo(AssignmentTimes)) {

System.out.println("Could not rename temporary file.");

}

deleteAssignmentError = false;

} catch (Exception e) {

e.printStackTrace();

System.out.println("An error has occurred while deleting an assignment. Please enter -1 to exit back to menu.");

}

} while (deleteAssignmentError);

System.out.println("Deleted assignment.");

}

public static void DisplayDayMethod() {

Scanner input = new Scanner(System.in);

DateTimeFormatter formatter = DateTimeFormatter.ofPattern("MMMM d, yyyy", Locale.ENGLISH);

ReadAssignmentsFile();

ReadAssignmentTimesFile();

String displayDay = null;

LocalDate formattedDisplayDay = null;

boolean dateError = true;

System.out.println("Please enter what day you would like to display (In the format Month Day, Year):");

do {

try {

displayDay = input.nextLine();

formattedDisplayDay = LocalDate.parse(displayDay, formatter);

if (displayDay.equals("-1")) {

System.out.println("Exiting to menu.");

return;

} else {

dateError = false;

}

} catch (Exception e) {

e.printStackTrace();

System.out.println("There was an error in your format. Please try again.");

}

} while (dateError);

if (assignmentDueDates.isEmpty()) {

System.out.println("There are no assignments on " + displayDay + ".");

} else {

System.out.println("\nThese are the assignments that are due on " + displayDay + ":");

for (int i = 0; i < assignmentDueDates.size(); i++) {

if (assignmentDueDates.get(i).equals(String.valueOf(formattedDisplayDay))) {

System.out.println(" - " + assignmentSubjects.get(i) + ": " + assignmentNames.get(i) + " is due on "

+ assignmentDueDates.get(i));

}

}

}

if (assignmentTimesDueDates.isEmpty()) {

System.out.println("There are no assignments to work on, on " + displayDay + ".");

} else {

System.out.println("\nThese are the assignments you have to work on, on " + displayDay + ":");

for (int i = 0; i < assignmentTimesDueDates.size(); i++) {

if (assignmentTimesDueDates.get(i).equals(String.valueOf(formattedDisplayDay))) {

System.out.println(" - Work on " + assignmentTimesSubject.get(i) + ": " + assignmentTimesName.get(i) + " for "

+ assignmentTimesHours.get(i) + " hours as it is due on " + assignmentTimesDueDates.get(i));

}

}

}

}

public static void DeleteAssignment() {

Scanner input = new Scanner(System.in);

//reads both Assignment files to ensure they are up to dare in case new assignments have been created after last read

ReadAssignmentsFile();

ReadAssignmentTimesFile();

//if no assignments (no names in assignment file) then there are no assignments created yet

if (assignmentNames.isEmpty()) {

System.out.println("There are no assignments created yet, so none can be deleted.");

return;

}

//print out all current assignments with their subject, name, and due date

System.out.println("Here are your current assignments:");

for (int i = 0; i < assignmentDueDates.size(); i++) {

System.out.println("Subject: '" + assignmentSubjects.get(i) + "' Name: '" + assignmentNames.get(i) + "' Due Date: '"

+ assignmentDueDates.get(i) + "'");

}

//prompts user to enter the name of one of the assignments they want to delete

System.out.println("Please enter the name of the assignment you want to delete, or -1 to exit back to menu:");

//String that is equal to the user's input

String deleteAssignment;

//boolean for error handling

boolean deleteAssignmentError = true;

do {

try {

boolean assignmentExists = false;

//keeps getting inputs until input is an assignment that exists, or -1 is used to exit back to menu

while (true) {

deleteAssignment = input.nextLine();

//for statement transverses every value in the assignment names arraylist

for (String assignmentName : assignmentNames) {

//if assignment inputted matches one in the assignmentName array, it exists

if (deleteAssignment.equals(assignmentName)) {

assignmentExists = true; //boolean variable to exit the loop

break;

}

}

if (deleteAssignment.equals("-1")) { //if input is -1, return to menu

return;

} else if (!assignmentExists) { //if inputted assignment does not exist, ask user to enter again

System.out.println("Please enter an assignment that exists.");

} else {

break;

}

}

//deletes inputted assignment from Assignment file

//creates temporary assignments file

try {

if (AssignmentsTemporary.createNewFile()) {

System.out.println("New temporary assignment file created.");

}

} catch (Exception e) {

System.out.println("An error occurred while creating a temporary assignment file.");

}

//arraylist with values copied from the Assignments file is made

Scanner assignmentsFileReader = new Scanner(new FileInputStream("Assignments.txt"));

ArrayList<String> assignmentFileArray = new ArrayList<>();

//every line in text file becomes an index in the arraylist

while (assignmentsFileReader.hasNext()) {

assignmentFileArray.add(assignmentsFileReader.nextLine());

}

assignmentsFileReader.close();

//finds the index value of the line with the same name as the assignment that needs to be deleted

int indexOfDeletedAssignment = 0;

//transverse assignmentFileArray searching for value that is equal to the input

for (int i = 0; i < assignmentFileArray.size(); i++) {

//if the assignment with the same name is found

if (assignmentFileArray.get(i).equals(deleteAssignment)) {

//mark the index down (+1 to match the line numbers of text files, arrays start at 0)

indexOfDeletedAssignment = i + 1;

}

}

//copies assignment file to temp assignment file minus the details for the deleted assignment

//if the line in the assignment file matches the lines which we have found match the assignment we want

// to delete, then don't copy it into the new file

int assignmentFileLineNumber = 1;

Scanner assignmentsFileReader2 = new Scanner(new FileInputStream("Assignments.txt"));

BufferedWriter assignmentsTempBw = new BufferedWriter(new FileWriter(AssignmentsTemporary));

PrintWriter assignmentsTempPw = new PrintWriter(assignmentsTempBw);

//while the file has more lines to read

while (assignmentsFileReader2.hasNext()) {

//three lines numbers containing information of the assignment needed to be deleted

//if the current line is not equal to one with information of deleted assignment, copy it into temporary file

if (assignmentFileLineNumber != indexOfDeletedAssignment - 1 &&

assignmentFileLineNumber != indexOfDeletedAssignment &&

assignmentFileLineNumber != indexOfDeletedAssignment + 1) {

assignmentsTempPw.println(assignmentsFileReader2.nextLine());

} else {

//else if it is equal, skip the line to remove the value by not copying into temp file

assignmentsFileReader2.nextLine();

}

//increment the line number

assignmentFileLineNumber++;

}

assignmentsTempBw.close();

assignmentsTempPw.close();

assignmentsFileReader2.close();

//deletes original assignment file with assignment we want to delete

if (!Assignments.delete()) {

System.out.println("Could not delete original assignment file.");

}

//renames the file with the deleted file removed to the original file

if (!AssignmentsTemporary.renameTo(Assignments)) {

System.out.println("Could not rename temporary file.");

}

//deletes inputted assignment from Assignment Times file

//creates temporary assignment times file

try {

if (AssignmentTimesTemporary.createNewFile()) {

System.out.println("New temporary assignment times file created.");

}

} catch (Exception e) {

System.out.println("An error occurred while creating the temporary assignment file.");

}

//reads the subject portion of the file, and if it is equal to the name of the assignment needed

//to be deleted, skip the next 4 lines as those all pertain to the same assignment

Scanner assignmentTimesFileReader = new Scanner(new FileInputStream("AssignmentTimes.txt"));

BufferedWriter assignmentTimesTempBw = new BufferedWriter(new FileWriter(AssignmentTimesTemporary));

PrintWriter assignmentTimesTempPw = new PrintWriter(assignmentTimesTempBw);

//while Assignment Times file has another line to be read

while (assignmentTimesFileReader.hasNext()) {

//String with value equal to next line in text file

String nextAssignmentTimesFileLine = assignmentTimesFileReader.nextLine();

//if the current line's value is equal to the name of the inputted assignment

if (nextAssignmentTimesFileLine.equals(deleteAssignment)) {

//skip the next three lines, which all contain information related to the inputted assignment

assignmentTimesFileReader.nextLine();

assignmentTimesFileReader.nextLine();

assignmentTimesFileReader.nextLine();

} else { //else print the current line and next three lines into the temp file

assignmentTimesTempPw.println(nextAssignmentTimesFileLine);

assignmentTimesTempPw.println(assignmentTimesFileReader.nextLine());

assignmentTimesTempPw.println(assignmentTimesFileReader.nextLine());

assignmentTimesTempPw.println(assignmentTimesFileReader.nextLine());

}

}

assignmentTimesTempBw.close();

assignmentTimesTempPw.close();

assignmentTimesFileReader.close();

//deletes original assignment times file with assignments we want to delete

if (!AssignmentTimes.delete()) {

System.out.println("Could not delete original assignment file.");

}

//renames the temporary assignment times file without deleted assignments to original file

if (!AssignmentTimesTemporary.renameTo(AssignmentTimes)) {

System.out.println("Could not rename temporary file.");

}

System.out.println("Deleted assignment.");

//terminates outside do-while loop so loop can be exited and can return to menu

deleteAssignmentError = false;

} catch (Exception e) { //handles any errors that may come up

System.out.println("An error has occurred while deleting an assignment. Please enter -1 to exit back to menu.");

}

} while (deleteAssignmentError); //if any invalid input is entered

System.out.println("Please enter a valid assignment.");

}

//reads assignment file and puts information into proper arrays

public static void ReadAssignmentsFile() {

try {

Scanner fileReader = new Scanner(new FileInputStream("Assignments.txt"));

//clear all arraylists from reading to prevent values from stacking (from multiple reads)

assignmentSubjects.clear();

assignmentNames.clear();

assignmentDueDates.clear();

//puts values in text file into their respective arraylist, values are printed in the order they are read, and information

//pertaining to the same assignment have the same index across all arrays (easily searched for and accessed with for loop)

while (fileReader.hasNext()) {

assignmentSubjects.add(fileReader.nextLine()); //subject of assignment that is due

assignmentNames.add(fileReader.nextLine()); //name of assignment that is due

assignmentDueDates.add(fileReader.nextLine()); //due date of assignment

}

fileReader.close();

} catch (IOException ex) {

System.out.println("Error while reading assignment file. Please try again.");

}

}

//creates the assignment file if one does not exist

public static void CreateAssignmentsFile() {

try {

if (Assignments.createNewFile()) {

System.out.println("New assignment file created.");

} else {

System.out.println("The assignment file already exists.");

}

} catch (Exception e) {

System.out.println("An error occurred while creating the assignment file. Please try again.");

e.printStackTrace();

}

}

//reads assignment times file and puts information into proper arrays

public static void ReadAssignmentTimesFile() {

try {

Scanner fileReader = new Scanner(new FileInputStream("AssignmentTimes.txt"));

//clear all arraylists from reading to prevent duplicate values (from multiple reads)

assignmentTimesName.clear();

assignmentTimesSubject.clear();

assignmentTimesHours.clear();

assignmentTimesDueDates.clear();

//while loop that continues if there is another line in the Assignment Times text file

while (fileReader.hasNext()) {

//puts values in text file into their respective arraylist, values are printed in the order they are read, and information

//pertaining to the same assignment have the same index across all arrays (easily searched for and accessed with for loop)

assignmentTimesName.add(fileReader.nextLine()); //name of the assignment that has to be worked on

assignmentTimesSubject.add(fileReader.nextLine()); //subject of assignment that has to be worked on

assignmentTimesHours.add(fileReader.nextLine()); //hours that the assignment has to be worked on for

assignmentTimesDueDates.add(fileReader.nextLine()); //day the assignment has to be worked on

}

fileReader.close();

} catch (IOException ex) {

System.out.println("Error while reading assignment times file. Please try again.");

}

}

//creates the assignment times file if one does not exist

public static void CreateAssignmentTimesFile() {

try {

if (AssignmentTimes.createNewFile()) {

System.out.println("New assignment times file created.");

} else {

System.out.println("The assignment times file already exists.");

}

} catch (Exception e) {

System.out.println("An error occurred while creating the assignment times file. Please try again.");

e.printStackTrace();

}

}

//returns assignments file arrays

public static ArrayList<String> GetAssignmentDueDates() {

return assignmentDueDates;

}

public static ArrayList<String> GetAssignmentNames() {

return assignmentNames;

}

public static ArrayList<String> GetAssignmentSubjects() {

return assignmentSubjects;

}

//returns assignment times file arrays

public static ArrayList<String> GetAssignmentTimesSubject() {

return assignmentTimesSubject;

}

public static ArrayList<String> GetAssignmentTimesName() {

return assignmentTimesName;

}

public static ArrayList<String> GetAssignmentTimesHours() {

return assignmentTimesHours;

}

public static ArrayList<String> GetAssignmentTimesDueDates() {

return assignmentTimesDueDates;

}

}