import java.io.\*;

import java.time.\*;

import java.time.format.DateTimeFormatter;

import java.util.\*;

public class User {

private static String *username*;

private static LocalDate *birthday*;

private static String *runStartupUpcomingAssignments*;

private static final File *UserFile* = new File("User.txt");

private static final File *UserTempFile* = new File("UserTemp.txt");

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

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

private static final ArrayList<LocalDate> *yearlyEventDates* = new ArrayList<>();

//creates all the files needed to run program, only called once in startup method as not needed after

public User(String name, LocalDate formattedBirthday) {

User.*username* = name;

User.*CreateUserFile*(formattedBirthday);

Games.*CreateGamesFile*();

Assignments.*CreateAssignmentsFile*();

Assignments.*CreateAssignmentTimesFile*();

}

public static void ChangeDayThreshold() throws IOException {

*ReadUserFile*();

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

System.*out*.println("Enter -1 to exit back to menu.");

System.*out*.println("Please input the new value for number of days you would like assignments to display in advance.\n" +

"(From 0, to any positive integer)");

int newDayThreshold = 0;

boolean inputError = true;

do {

try {

newDayThreshold = input.nextInt();

if (newDayThreshold == -1) {

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

return;

} else if (newDayThreshold < 0) {

System.*out*.println("Day threshold cannot be negative. Please enter another value.");

} else {

inputError = false;

}

} catch (Exception e) {

System.*out*.println("Please enter an integer value, as days must be whole numbers.");

}

} while (inputError);

Scanner userFileReader = new Scanner(new FileInputStream("User.txt"));

BufferedWriter userBw = new BufferedWriter(new FileWriter(*UserTempFile*));

PrintWriter userPw = new PrintWriter(userBw);

//creates temporary games file

try {

if (*UserTempFile*.createNewFile()) {

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

}

} catch (Exception e) {

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

}

userPw.println(userFileReader.nextLine());

userFileReader.nextLine();

userPw.println(newDayThreshold);

while (userFileReader.hasNext()) {

userPw.println(userFileReader.nextLine());

}

userFileReader.close();

userBw.close();

userPw.close();

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

if (!*UserFile*.delete()) {

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

}

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

if (!*UserTempFile*.renameTo(*UserFile*)) {

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

}

System.*out*.println("New number of days assignments are displayed in advance set.");

}

public static void CreateYearlyEvent() {

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

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

System.*out*.println("Enter -1 at any time to exit back to the menu and terminate current process.");

String eventName;

while (true) {

System.*out*.println("Please enter a descriptive name for the yearly event (Maximum of 13 characters):");

eventName = input.nextLine();

if (eventName.length() > 13) {

System.*out*.println("Name for yearly event cannot exceed 13 characters! Please try something shorter.");

} else if (eventName.equals("-1")) {

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

return;

} else {

break;

}

}

System.*out*.println("Please enter a description for the yearly event:");

String eventDescription = input.nextLine();

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

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

return;

}

String assignmentDueDate;

LocalDate formattedEventDate = null;

boolean dateError = true;

System.*out*.println("Please enter the date of the yearly event (In the format Month Day):");

do {

try {

assignmentDueDate = input.nextLine();

assignmentDueDate = assignmentDueDate.concat(", 2000");

formattedEventDate = LocalDate.*parse*(assignmentDueDate, formatter);

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

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

return;

} else {

dateError = false;

}

} catch (Exception e) {

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

}

} while (dateError);

System.*out*.println("Information input complete! Created new yearly event.");

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

BufferedWriter bw = new BufferedWriter(fw);

PrintWriter pw = new PrintWriter(bw)) {

pw.println(eventName + "\n" + eventDescription + "\n" + formattedEventDate);

} catch (Exception e) {

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

}

input.close();

User.*ReadUserFile*();

Calendars.*PrintCalendar*();

}

public static void DeleteYearlyEvent() {

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

*ReadUserFile*();

if (*yearlyEventNames*.isEmpty()) {

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

return;

}

System.*out*.println("Here is the list of events you have right now:");

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

System.*out*.println("Name: '" + *yearlyEventNames*.get(i) + "' Description: '" + *yearlyEventDescriptions*.get(i)

+ "' Date: '" + *yearlyEventDates*.get(i) + "'");

}

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

String deleteEvent;

boolean deleteEventError = true;

do {

try {

boolean eventExists = false;

while (true) {

deleteEvent = input.nextLine();

for (String yearlyEventName : *yearlyEventNames*) {

if (deleteEvent.equals(yearlyEventName)) {

eventExists = true;

break;

}

}

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

return;

} else if (!eventExists) {

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

} else {

break;

}

}

//creates temporary assignment file

try {

if (*UserTempFile*.createNewFile()) {

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

}

} catch (Exception e) {

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

}

//array with exact values from assignment file

Scanner userFileReader = new Scanner(new FileInputStream("User.txt"));

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

while (userFileReader.hasNext()) {

userFileArray.add(userFileReader.nextLine());

}

userFileReader.close();

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

int indexOfDeletedEvent = 0;

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

if (userFileArray.get(i).equals(deleteEvent)) {

indexOfDeletedEvent = 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 userFileLineNumber = 1;

Scanner eventsFileReader2 = new Scanner(new FileInputStream("User.txt"));

BufferedWriter eventsTempBw = new BufferedWriter(new FileWriter(*UserTempFile*));

PrintWriter eventsTempPw = new PrintWriter(eventsTempBw);

while (eventsFileReader2.hasNext()) {

if (userFileLineNumber != indexOfDeletedEvent &&

userFileLineNumber != indexOfDeletedEvent + 1 &&

userFileLineNumber != indexOfDeletedEvent + 2) {

eventsTempPw.println(eventsFileReader2.nextLine());

} else {

eventsFileReader2.nextLine();

}

userFileLineNumber++;

}

eventsTempPw.close();

eventsFileReader2.close();

eventsTempBw.close();

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

if (!*UserFile*.delete()) {

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

}

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

if (!*UserTempFile*.renameTo(*UserFile*)) {

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

}

deleteEventError = false;

} catch (Exception e) {

e.printStackTrace();

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

}

} while (deleteEventError);

System.*out*.println("Yearly event deleted.");

}

public void CalculateAge(LocalDate formattedBirthday) {

System.out.println("Your birthday is on " + formattedBirthday);

System.out.println("So you're " + Period.between(formattedBirthday, LocalDate.now()).getYears() + " years old!");

}

//creates user file and writes username, variable to tell program has been run before, and user birthday

private static void CreateUserFile(LocalDate formattedBirthday) {

try {

if (UserFile.createNewFile()) {

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

} else {

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

}

} catch (Exception e) {

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

e.printStackTrace();

}

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

BufferedWriter bw = new BufferedWriter(fw);

PrintWriter pw = new PrintWriter(bw)) {

pw.println(username + "\n" + "1\n" + formattedBirthday);

pw.println("Christmas Eve\nChristmas is coming tomorrow!!\n2000-12-24");

pw.println("Christmas Day\nMerry Christmas!!! Hope you have a neat day! :D\n2000-12-25");

pw.println("Family Day\nToday is family day! Spend some time with your family! :D\n2000-02-15");

} catch (Exception e) {

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

}

}

//reads the user.txt and puts values into arrays

public static void ReadUserFile() {

try {

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

username = fileReader.nextLine();

runStartupUpcomingAssignments = String.valueOf(fileReader.nextLine());

birthday = LocalDate.parse(fileReader.nextLine());

yearlyEventNames.clear();

yearlyEventDescriptions.clear();

yearlyEventDates.clear();

while (fileReader.hasNext()) {

yearlyEventNames.add(fileReader.nextLine());

yearlyEventDescriptions.add(fileReader.nextLine());

yearlyEventDates.add(LocalDate.parse(fileReader.nextLine()));

}

fileReader.close();

} catch (Exception e) {

System.out.println("No User file exists yet. Running startup method.");

}

}

public static void DeleteUserFile() {

if (!UserFile.delete()) {

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

} else {

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

}

}

//returns true if program has not been run before, if it has the program puts 0 into the user file

public static boolean CheckRunStartup() {

boolean runStart = true;

try {

if (!runStartupUpcomingAssignments.isEmpty()) {

runStart = false;

}

return runStart;

} catch (Exception e) {

return true;

}

}

public static String GetRunStartupUpcomingAssignments() {

return runStartupUpcomingAssignments;

}

public static String GetUsername() {

return username;

}

public static LocalDate GetBirthday() {

return birthday;

}

public static ArrayList<String> GetYearlyEventNames() {

return yearlyEventNames;

}

public static ArrayList<String> GetYearlyEventDescriptions() {

return yearlyEventDescriptions;

}

public static ArrayList<LocalDate> GetYearlyEventDates() {

return yearlyEventDates;

}

}