```java
/**
 *******************************************************************************
 *                                                                             *
 ******************************** FINAL VERSION ********************************
 *                                                                             *
 * PROGRAM: BLOOD TRANSFUSION MANAGER                                          *
 *                                                                             *
 * AUTHOR: ALEKSANDAR TSANKOV MLADENOV                                         *
 *                                                                             *
 * STUDENT NUMBER: 2976196                                                     *
 *                                                                             *
 * *****************************************************************************
 * */

import java.util.*;
import java.io.*;
import java.text.SimpleDateFormat;


public class BloodTransfusionManager {


    // below takes a digit and converts to its blood type equivalent

    public static String convertDigitToBloodType(int a) {
        String b = "";

        switch (a) {
        case 0:
            b = "O-";
            break;
        case 1:
            b = "O+";
            break;
        case 2:
            b = "A-";
            break;
        case 3:
            b = "A+";
            break;
        case 4:
            b = "B-";
            break;
        case 5:
            b = "B+";
            break;
        case 6:
            b = "AB-";
            break;
        case 7:
            b = "AB+";
            break;
        }
        return b;
```

```java
}

// will convert blood type to digit for array manipulation

public static int convertBloodTypeToDigit(String a) {

    int b = 0;

    if (a.equalsIgnoreCase("O-"))
        b = 0;
    if (a.equalsIgnoreCase("O+"))
        b = 1;
    if (a.equalsIgnoreCase("A-"))
        b = 2;
    if (a.equalsIgnoreCase("A+"))
        b = 3;
    if (a.equalsIgnoreCase("B-"))
        b = 4;
    if (a.equalsIgnoreCase("B+"))
        b = 5;
    if (a.equalsIgnoreCase("AB-"))
        b = 6;
    if (a.equalsIgnoreCase("AB+"))
        b = 7;

    return b;
}

// will brake entry in to blood type only

public static String getBlood(String a) {
    String entry = a;
    String bloodType;
    int index = entry.indexOf(";") + 1;
    bloodType = entry.substring(index).trim().replace(" ", "");
    return bloodType;
}

// will brake the entry in to name only

public static String getName(String a) {
    String entry = a;
    String name;
    int index = entry.indexOf(";");
    name = entry.substring(0, index);
    return name;
}

// will give entries a more user friendly readable formating

public static String formatEntry(String a) {
    String entry = a;
    String formated;
    formated = entry.replace(";", ": Blood Type (") + ").";
```

```
        return formated;
    }

    public static void main(String[] args) throws Exception {
        // TODO Auto-generated method stub

        // ------------- Main method variables --------------//

        Scanner input = new Scanner(System.in);
        String recepientName, donorName, bloodType, compatibility = " test ";
        char choice;
        int entryCount = 1, sortArraysCounter = 0; // used to count entries
        Boolean terminate = false;
        SimpleDateFormat dateFormat = new SimpleDateFormat(" k:mm E d-M-yyyy ");
        Calendar calendar = new GregorianCalendar();
        File appointmentsTXT = new File("appointments.txt"); // set up files
        File checkLog = new File("checklog.txt");
        FileWriter fileWriter = new FileWriter(appointmentsTXT); // set up file writer to memory
        PrintWriter printWriter = new PrintWriter(fileWriter); // set up print writer to file

        // ------------- Main method variables --------------//

        // Boolean array to check rec/donor copablity //

        boolean copabilityTable[][] =

                {
                /*------------------------------------------------------------*/
                                  //DON
                        //REC     /* O-    O+     A-     A+      B-     B+   AB-   AB+ */
                        /* O- */ { true, false, false, false, false, false, false, false },
                        /*------------------------------------------------------------*/
                        /* O+ */ { true, true, false, false, false, false, false, false },
                        /*------------------------------------------------------------*/
                        /* A- */ { true, false, true, false, false, false, false, false },
                        /*------------------------------------------------------------*/
                        /* A+ */ { true, true, true, true, false, false, false, false },
                        /*------------------------------------------------------------*/
                        /* B- */ { true, false, false, false, true, false, false, false },
                        /*------------------------------------------------------------*/
                        /* B+ */ { true, true, false, false, true, true, false, false },
                        /*------------------------------------------------------------*/
                        /* AB- */ { true, false, true, false, true, false, true, false },
                        /*------------------------------------------------------------*/
                        /* AB+ */ { true, true, true, true, true, true, true, true }

                /*------------------------------------------------------------*/

                };

        // Blood types for 1-1 manul comparison

        String bloodTypes[] = { "O-", "O+", "A-", "A+", "B-", "B+", "AB-", "AB+" };
        // Array lists for matching and apoitments

        ArrayList<String> recList = new ArrayList<String>(); // Used to store valid entries from reading the rec file

        ArrayList<String> donList = new ArrayList<String>(); // Used to store valid entries from reading the rdoc file

        ArrayList<String> recOptions = new ArrayList<String>(); // Temporary Storage of Transfusion options for 1-1
                                                                // check
        ArrayList<String> donOptions = new ArrayList<String>(); // Temporary Storage of Transfusion options for 1-1
                                                                // check (Not in use for the moment)
        ArrayList<Integer> recOptions1 = new ArrayList<Integer>(); // Temporary Storage of Transfusion options for total number of opt
ions
                                                                    // check
        ArrayList<Integer> donOptions1 = new ArrayList<Integer>(); // Temporary Storage of Transfusion options for total number of opt
ions
                                                                    // check (Not in use for the moment)
        ArrayList<String> recListSorted = new ArrayList<String>(); // used to Sort the above recList in to blood types
                                                                    // with the ones with less options on top and the one with more on
 the bottom
        ArrayList<String> donListSorted = new ArrayList<String>(); // used to Sort the above docList in to blood types
                                                                    // with the ones with less options on top and theone with more on
the bottom
        ArrayList<String> appointments = new <String> ArrayList(); // used to store final appointments

        ArrayList<String> recOverflow = new ArrayList<String>(); // will repopulate sorted list with excess rec Options
                                                                    // if any
        ArrayList<String> donOverflow = new ArrayList<String>();// will repopulate sorted list with excess don Options
                                                            // if any
        ArrayList<String>[] appointmentsCalendar = new ArrayList [260]; // ArrayList of Arrays used to store appointmetns

        for (int i = 0; i < appointmentsCalendar.length; i++) { // for Loop to initialise Array of ArrayList
            appointmentsCalendar[i] = new ArrayList<String>();
        }

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

        BufferedReader reader = null; // used to read files set as null for multyple use

        // try catch block is looking for missing file error if any are found a Boolean
        // is trigered to lock the rest of the program

        try {

            String line;
            entryCount = 1;

            // Below loop will read the recipients list and it will ignore errors such as
            // spaces in the midle or around blood types but if anything else is found it
            // will mark the error and ignore it.

            System.out.println(
                    "\n----------------------------------------READING RECIPIENTS----------------------------------------");

            reader = new BufferedReader(new FileReader("recipients.txt"));
```

```
while ((line = reader.readLine()) != null) {

    String validityCheck = line;// find the blood type , trim and replace all spaces
    int index = validityCheck.indexOf(";") + 1;
    bloodType = validityCheck.substring(index).trim().replace(" ", "");

    for (int i = 0; i < bloodTypes.length; i++) { // checks if the blood type is valid if it is adds, if its
                                                    // not it will discard it

        if (bloodType.equalsIgnoreCase(bloodTypes[i])) {
            System.out.println(entryCount + " " + line.substring(0, (index - 1)));
            recList.add(line);
            break;
        } else if (i == 7) {
            System.out.println("Warning entry " + "(" + line
                    + ") blood type could not be verified or added to registry!");
            entryCount--;
        }
    }
    entryCount++;// counter for final entries added
}
reader.close();// closes the reader

// Below loop will read the donor list and it will ignore errors such as spaces
// in the midle or around blood types but if anything else is found it will mark
// the error and ignore it.
System.out.println(
        "------------------------------------------READING DONORS-----------------------------------------");
entryCount = 1;
reader = new BufferedReader(new FileReader("donors.txt"));
while ((line = reader.readLine()) != null) {

    String validityCheck = line; // find the blood type , trim and replace all spaces
    int index = validityCheck.indexOf(";") + 1;
    bloodType = validityCheck.substring(index).trim().replace(" ", "");

    for (int i = 0; i < bloodTypes.length; i++) { // checks if the blood type is valid if it is adds, if its
                                                    // not it will discard it

        if (bloodType.equalsIgnoreCase(bloodTypes[i])) {
            System.out.println(entryCount + " " + line.substring(0, (index - 1)));
            donList.add(line);
            break;
        } else if (i == 7) {
            System.out.println("Warning entry " + "(" + line
                    + ") blood type could not be verified or added to registry!");
            entryCount--;
        }
    }
    entryCount++; // counter for final entries added
}
reader.close(); // closes the reader

// if any files are missing it will automaticaly close the program by switching
```

```
    // the terminate Boolean

} catch (Exception e) {
    terminate = true;
    System.out.println("Error critical file not found");
    System.out.println("System Error: Terminating ...");

}

// if terminate isnt trigered it will initialise below program
if (terminate == false) {
// Prints the final lists if compiled corectly

System.out.println("******************************************************");
System.out.println("\n----------------------FINAL RECEPIENTS LIST----------------------");
for (int i = 0; i < recList.size(); i++) {
    System.out.println((i) + 1 + ". " + formatEntry(recList.get(i)));
}
System.out.println("\n------------------------FINAL DONORS LIST------------------------");
for (int i = 0; i < donList.size(); i++) {
    System.out.println((i) + 1 + ". " + formatEntry(donList.get(i)));
}

    System.out.println("\n*******************************************************");
    System.out.println("Would you like to evaluate donors and recepients individualy ? Y/N");
    choice = choice = input.next().charAt(0);
    while (choice == 'Y' || choice == 'y') {
        int recepientNumber = 0, donorNumber = 0;
        System.out.println("\n******************************************************");
        System.out.println("Please choose recepient for donor match assesment?");
        System.out.println("------------------------------------------------------");
        System.out.println("1.Recepient ( enter number of chosen recepient field )");
        recepientNumber = input.nextInt() - 1;
        recepientName = recList.get(recepientNumber);
        System.out.println("Recepient- " + formatEntry(recepientName));
        System.out.println("\n------------------------------------------------------");
        System.out.println("2.Donor ( enter number of chosen donor field )");
        donorNumber = input.nextInt() - 1;
        donorName = donList.get(donorNumber);
        System.out.println("Donor- " + formatEntry(donorName));

        for (int i = 0; i < copabilityTable[convertBloodTypeToDigit(getBlood(recepientName))].length; i++) {

            if (copabilityTable[convertBloodTypeToDigit(getBlood(recepientName))][i] == true) {
                recOptions.add(convertDigitToBloodType(i));
            }
        }

        for (int i = 0; i < recOptions.size(); i++) {
            compatibility = " IS NOT COMPATIBLE WITH ";
            if (recOptions.get(i).equalsIgnoreCase(getBlood(donorName))) {
                compatibility = " IS COMPATIBLE WITH ";
                break;
            }
            }

            System.out.println("\n-----------------------------------------------------------");
            System.out.println("Recepient- " + formatEntry(recepientName));
            System.out.println("Donor- " + formatEntry(donorName));
            System.out.println(
                    "\n" + getName(recepientName) + compatibility + getName(donorName) + " for transfusion");
            System.out.println(getName(recepientName) + " can take transufions from blood types " + recOptions);
            System.out.println(getName(donorName) + " has blood type " + getBlood(donorName));
            System.out.println("\nAlternative Donors for " + getName(recepientName)
                    + "\n-----------------------------------------------------------");
            for (int i = 0; i < donList.size(); i++) {
                String name = donList.get(i);
                for (int j = 0; j < recOptions.size(); j++) {
                    if (recOptions.get(j).equalsIgnoreCase(getBlood(name))) {
                        System.out.println(formatEntry(name));
                    }
                }
            }
            recOptions.clear();
            System.out.println("Would you like to assess another patient Y/N ?");
            choice = input.next().charAt(0);

        }

        // the below loop transfers from original aray to new one

        while (recList.size() != recListSorted.size()) {
            for (int i = 0; i < recList.size(); i++) {
                String a = recList.get(i);
                recListSorted.add(a);
            }
            sortArraysCounter++;
        }
        sortArraysCounter = 0;

        // the below loop transfers from original aray to new one

        while (donList.size() != donListSorted.size()) {
            for (int i = 0; i < donList.size(); i++) {
                String a = donList.get(i);
                donListSorted.add(a);
            }
            sortArraysCounter++;
        }
        sortArraysCounter = 0;

// ---------------------old loop-----------------------

//      while (donList.size() != donListSorted.size()) {
//          for (int i=0;i<donList.size();i++) {
//              String a=donList.get(i);
//              String b=getBlood(a);
//              int c = convertBloodTypeToDigit(b);
//              if (c == sortArraysCounter) {
//                  donListSorted.add(a);
//              }
//          }
//          sortArraysCounter++;
//      }
//      sortArraysCounter=0;

// ---------------------old loop-----------------------

            int count = 0;
            // check every posible transfusion option and reorganise rec lists

            fileWriter = new FileWriter(checkLog);
            printWriter = new PrintWriter(fileWriter);
            int checkEntry = 0;

            // this loop goes through every posible donor and every posible recipient and check if they are compatible or not
            // if they are compatile after cheking the copability table it records the results an takes them in to account


            printWriter.println("***************************** RECIPIENT CHECK *******************************\n");
            for (int i = 0; i < recListSorted.size(); i++) {
                for (int j = 0; j < donListSorted.size(); j++) {
                    String recEntry = recListSorted.get(i); // entry from ArrayList
                    String recBloodType = getBlood(recEntry);
                    int recBloodNumber = convertBloodTypeToDigit(recBloodType); // convert to digit
                    String donEntry = donListSorted.get(j); // entry from ArrayList
                    String donBloodType = getBlood(donEntry);
                    int donBloodNumber = convertBloodTypeToDigit(donBloodType); // convert to digit
                    Boolean match = copabilityTable[recBloodNumber][donBloodNumber];
                    checkEntry++;

                    if (match == true) {
                        donOptions.add(donEntry);
                        count++;
                    }

                    printWriter.println("\n-------------------------------------------------------\n");
                    printWriter.println("-------------------------(" + checkEntry + ")-----------------------");
                    printWriter.println("\nRecipient: " + formatEntry(recEntry));
                    printWriter.println("\nDonor: " + formatEntry(donEntry));
                    printWriter.println("\nBlood type compatible " + "( " + match + " )");
                    printWriter.println("\nPosible options for " + getName(recEntry) + " " + "( " + count + " )\n");

                    if (j + 1 == donListSorted.size()) {
                        recOptions1.add(count);
                        for (int k = 0; k < donOptions.size(); k++) {
                            printWriter.println(formatEntry(donOptions.get((k))));
                        }
                        donOptions.clear();
                        count = 0;
                    }
                }
            }

            // ---------------------TESTING ARRAYS---------------------

            System.out.println("\n---------recListUnSorted (Sorthing visualisation)-----------\n");
            for (int i = 0; i < recListSorted.size(); i++) {
                System.out.println(recListSorted.get(i) + "   " + recOptions1.get(i));
            }
            System.out.println("-----------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------");
            System.out.println(recOptions1);
            System.out.println("-----------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------");

            // ---------------------TESTING ARRAYS---------------------

            // Below loop will organise posible options starting with lower numbers to teh front and people with more options to the b
ottom
            // it will also move the coresponding recipients in acordance with the blood option move keeping them in harmony together

            for (int i = 0; i < recOptions1.size(); i++) {
                for (int j = 0; j < recOptions1.size(); j++) {
                    int a = recOptions1.get(i); // 1
                    int b = recOptions1.get(j); // 2
                    String c = recListSorted.get(i); // 1
                    String d = recListSorted.get(j); // 2
                    if (recOptions1.get(i) < recOptions1.get(j)) {
                        recOptions1.set(i, b); // i = j
                        recOptions1.set(j, a); // j = i
                        recListSorted.set(i, d); // i = j
                        recListSorted.set(j, c); // j = i

                    }
                }
            }

            // ---------------------TESTING BLOCK---------------------
            System.out.println(recOptions1);
            System.out.println("-----------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------");
            for (int i = 0; i < recListSorted.size(); i++) {
                System.out.println(recListSorted.get(i) + "   " + recOptions1.get(i));
            }
            // ---------------------TESTING BLOCK---------------------

            // check every posible transfusion option and reorganise dont lists
            for (int z = 0; z < 60; z++) {
                printWriter.println(
                        "**********************************************************************************************************");
                if (z == 29) {
                    printWriter.println(
                            "\n********************************************** DONORS CHECK ********************************************
***\n");
```

```
            }
        }


        // this loop goes through every posible donor and every posible recipient and check if they are compatible or not
        // if they are compatile after cheking the copability table it records the results an takes them in to account


        for (int i = 0; i < donListSorted.size(); i++) {
            for (int j = 0; j < recListSorted.size(); j++) {
                String donEntry = donListSorted.get(i); // entry from ArrayList
                String donBloodType = getBlood(donEntry);
                String recEntry = recListSorted.get(j); // entry from ArrayList
                String recBloodType = getBlood(recEntry);
                int recBloodNumber = convertBloodTypeToDigit(recBloodType); // convert to digit
                int donBloodNumber = convertBloodTypeToDigit(donBloodType); // convert to digit
                Boolean match = copabilityTable[recBloodNumber][donBloodNumber];
                checkEntry++;

                if (match == true) {
                    recOptions.add(recEntry);
                    count++;
                }

                printWriter.println("\n--------------------------------------------------------\n");
                printWriter.println("-------------------------(" + checkEntry + ")------------------------");
                printWriter.println("\nDonor: " + formatEntry(donEntry));
                printWriter.println("\nRecipient: " + formatEntry(recEntry));
                printWriter.println("\nBlood type compatible " + "( " + match + " )");
                printWriter.println(
                        "\nPosible options (to give blood) for " + getName(donEntry) + " " + "( " + count + " )\n");

                if (j + 1 == recListSorted.size()) {
                    donOptions1.add(count);
                    for (int k = 0; k < recOptions.size(); k++) {
                        printWriter.println(formatEntry(recOptions.get((k))));
                    }
                    recOptions.clear();
                    count = 0;
                }

            }
        }

        // ---------------------TESTING ARRAYS---------------------

        System.out.println("\n---------donListUnSorted (Sorthing visualisation)-----------\n");
        for (int i = 0; i < donListSorted.size(); i++) {
            System.out.println(donListSorted.get(i) + "  " + donOptions1.get(i));
        }
        System.out.println("-----------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------");
        System.out.println(donOptions1);
```

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

            // ---------------------TESTING ARRAYS---------------------

            // Below loop will organise posible options starting with lower numbers to teh front and people with more options to the b
ottom
            // it will also move the coresponding recipients in acordance with the blood option move keeping them in harmony together

            for (int i = 0; i < donOptions1.size(); i++) {
                for (int j = 0; j < donOptions1.size(); j++) {
                    int a = donOptions1.get(i); // 1
                    int b = donOptions1.get(j); // 2
                    String c = donListSorted.get(i); // 1
                    String d = donListSorted.get(j); // 2
                    if (donOptions1.get(i) < donOptions1.get(j)) {
                        donOptions1.set(i, b); // i = j
                        donOptions1.set(j, a); // j = i
                        donListSorted.set(i, d); // i = j
                        donListSorted.set(j, c); // j = i

                    }
                }
            }

            printWriter.close();

            System.out.println(donOptions1);
            System.out.println("------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------");

            // ---------------------TESTING BLOCK---------------------
            for (int i = 0; i < donListSorted.size(); i++) {
                System.out.println(donListSorted.get(i) + "  " + donOptions1.get(i));
            }
            // ---------------------TESTING BLOCK---------------------

            // Bellow for loop will populate appoitmetns for the next year with given criteria
            int testCounter = 1;
            int index = 0;

            calendar.set(Calendar.HOUR_OF_DAY, 8);
            calendar.set(Calendar.MINUTE, 0);
            calendar.add(Calendar.DATE, 2);

            for (int i=0; i< appointmentsCalendar.length;i++) {
                for(int j=0; j< 12 ;j++) {

            if ((calendar.get(Calendar.HOUR_OF_DAY) > 19)) {
                calendar.add(Calendar.DATE, 1);
                calendar.set(Calendar.HOUR_OF_DAY, 8);
            }
            if ((calendar.get(Calendar.DAY_OF_WEEK) == 6) // Friday
                    && (calendar.get(Calendar.HOUR_OF_DAY) > 19)) {
```

```
                calendar.add(Calendar.DATE, 3);
            }
            if (calendar.get(Calendar.DAY_OF_WEEK) == 7) { // Saturday
                calendar.add(Calendar.DATE, 2);
            }
            if (calendar.get(Calendar.DAY_OF_WEEK) == 1) { // Sunday
                calendar.add(Calendar.DATE, 1);
            }

            appointmentsCalendar[i].add(dateFormat.format(calendar.getTime()));
            calendar.add(Calendar.HOUR_OF_DAY, 1);

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



            /* Below loops are used to match people for appoitmetns while the recipient list is full with even a single entry.
             * Then a second look will match people betwean 2 lists if the recipients run out and all have matches then the loop will
break.
             * However if recipients remain they it will reload the donors again and book apoitments in 56 days if the donors reramin
however the loop will brake.
             * It will double check if donors can match an and stores them in an appoitment ArrayList with the first entry being an re
cipient the second a donor and the 3rd an date.*/
            int count1=0;
            Boolean donorDelay = false;
            while ((recListSorted.size() > 0)) {
                while ((recListSorted.size() > 0) && (donListSorted.size() > 0)) {
                    index = 0;
                    for (int j = 0; j < donListSorted.size(); j++) {

                        String recEntry = recListSorted.get(index); // entry from ArrayList
                        String recBloodType = getBlood(recEntry);
                        int recBloodNumber = convertBloodTypeToDigit(recBloodType); // convert to digit
                        String donEntry = donListSorted.get(j); // entry from ArrayList
                        String donBloodType = getBlood(donEntry);
                        int donBloodNumber = convertBloodTypeToDigit(donBloodType); // convert to digit
                        Boolean match = copabilityTable[recBloodNumber][donBloodNumber];
                        testCounter++;

                        // Once the blood types are reconfirmed a date is generated and if the donor has given blood already 56 days a
re added

                        int donorDelay1=0;

                        if (match == true) {
                            appointments.add(recEntry);
                            appointments.add(donEntry);
                            if (donorDelay == true) {
                                for (int z = appointments.size() - 3; z >= 0; z--) {
                                    if (donEntry.equals(appointments.get(z))) {
                                        int previousDate=Integer.parseInt(appointments.get(z+1));
                                        donorDelay1=40+previousDate;
                                        break;
                                    }
                                }
                            }

                            for ( int i = (0+donorDelay1) ; i < appointmentsCalendar.length; i++) {

                                if (appointmentsCalendar[i].size() != 0) {
                                    int array = i;
                                    int index1 = 0;

                                    while ((appointmentsCalendar[array].size() != 0)) {
                                        String day = Integer.toString(i);
                                        appointments.add(day);
                                        appointments.add(appointmentsCalendar[array].get(index1));
                                        appointmentsCalendar[array].remove(index1);
                                        break;
                                    }
                                    donorDelay1=0;
                                    break;
                                }

                            }
                            donOverflow.add(donEntry);
                            recListSorted.remove(recEntry);
                            donListSorted.remove(donEntry);
                            break;
                        }
                        if (j + 1 == donListSorted.size()) {
                            recOverflow.add(recEntry);
                            recListSorted.remove(recEntry);
                        }

                    }

                }

                if (donListSorted.size() == 0) {
                    donorDelay = true;
                }

                /* below 2 loops store overflow from both lists if we have either recepients or donors wich are unmatched
                they are added to an overflow and then the main lists are repopulated based on previous matching witout sorting by opt
ions this time.
                Now they are populating the main ArrayList on top people who were unmatched last time will remain in both lists on top
*/

                for (int i = 0; i < recOverflow.size(); i++) {
                    recListSorted.add(recOverflow.get(i));
                    if (i + 1 == recOverflow.size()) {
                        recOverflow.clear();
                    }
                }

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

```java
                    donListSorted.add(donOverflow.get(i));
                    if (i + 1 == donOverflow.size()) {
                        donOverflow.clear();
                    }
                }
            }
            System.out.println("*******************************************************************");
            System.out.println(donDelay);

            /* The Below loop prints the apoitments on the screen and then prints it to a text file.
             * As all teh information is stored on a single ArrrayList instead of using 3 separate loops i used one with modulo option
s. */
            fileWriter = new FileWriter(appointmentsTXT);
            printWriter = new PrintWriter(fileWriter);
            System.out.println("\n*********** appointments ***********\n");
            printWriter.println("\n*********** appointments ***********\n");
            System.out.println("--------------------------");

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

                if ((i + 1) % 4 == 1) {
                    String fRecipientName = getName(appointments.get(i));
                    String fRecipientBloodType = getBlood(appointments.get(i));
                    System.out.println(
                            "Recipient: < " + fRecipientName + " >    Blood Type: < " + fRecipientBloodType + " >");
                    printWriter.println(
                            "Recipient: < " + fRecipientName + " >    Blood Type: < " + fRecipientBloodType + " >");
                }

                if ((i + 1) % 4 == 2) {
                    String fDonorName = getName(appointments.get(i));
                    String fDonorBloodType = getBlood(appointments.get(i));
                    System.out.println(
                            "Donor: < " + fDonorName + " >            Blood Type: < " + fDonorBloodType + " >");
                    printWriter.println(
                            "Donor: < " + fDonorName + " >            Blood Type: < " + fDonorBloodType + " >");
                }

                if ((i + 1) % 4 == 0) {
                    System.out.println("Date: < " + appointments.get(i) + " >");
                    printWriter.println("Date: < " + appointments.get(i) + " >");
                    System.out.println("-------------------------------------------------------------------");
                    printWriter.println("-------------------------------------------------------------------");
                }
            }
            printWriter.close();
        }
    }
}
```