```
/*
Student name: Joaquin De Losada
Student ID: 6778757
Professor: Yifang Li
*/
public class Main {

//readdata, trickleDown, buildHeap, removeMin, inordertraverse, heapSort
   public static void main(String[] args) {
       DrugHeap db = new DrugHeap();

       db.ReadData();
       db.BuildHeap();
       db.InOrderTraverse();
       db.HeapSort();
   }
}
import java.io.*;
import java.lang.Math;
import java.nio.BufferUnderflowException;
import java.util.ArrayList;

public class DrugHeap {
   int currentSize; //Counter used to create the patients array.
```

```
                File writeToInOrderTraverse = new File("recourses//sockedApprovedInOrder.tab");

                File writeToHeapSorted = new File("recourses//dockedApprovedSorted.tab");


        public BufferedWriter writeFileInOrder, writeHeapSorted;


        public Drug root;


        public ArrayList<String> drugList = new ArrayList<>();


        public Drug[] drugArray, buildHeapArray, heapSorted;



        //Will read the data from the file and place it in a array list
        public void ReadData() {
              File txtFile = new File("recourses//dockedApproved.tab"); //Gets the file
            BufferedReader readFile;


            int totalArrayLength = 0;


            try{ //Try this code
                 readFile = new BufferedReader(new FileReader(txtFile)); //Read file


                String line = ""; //String that will contain each line
                //ArrayList<String> drugList = new ArrayList<>();
```

```
                //System.out.println(readFile);

                while (line != null) { //Loop through the txt file

                    line = readFile.readLine(); //Read the line

                    if (line == null) {

                        break;

                    } //If the current line is null it will stop reading

                      if (line.contains("Generic Name")) {} //If the line ever
contains name then it will do nothing. This is used for the first line of the
txt which dosent contain a patient

                          else { //Will add to the current patients list and
increase size of patients array for later

                        totalArrayLength++;

                        drugList.add(line);

                    }

                }

                 drugArray = new Drug[totalArrayLength]; //Create the array of
ADT of drug of the size of patients

                 drugList.forEach((i) ->{ //Go through each arraylist of drug
string to split the string up and add it to a drug data type

                      String[] currentPatient = i.split("\\t"); //Spliting the
drug into an array

                    Drug newDrug = new Drug(); //Create a drug data type

                     newDrug.SetGenericName(currentPatient[0]); //Set the drug
name

                         newDrug.SetSmiles(currentPatient[1]); //Set the drug
smiles

                      newDrug.SetDrugBankID(currentPatient[2]); //Set the drug
ID

                    newDrug.SetURL(currentPatient[3]); //Set the drug URL

                      newDrug.SetDrugGroup(currentPatient[4]); //Set the drugs
group
```

```
                    newDrug.SetScore(currentPatient[5]); //Set the drugs score

                      drugArray[drugList.indexOf(i)] = newDrug; //Add the drug
to the array of drugs

                });

            } catch (IOException e) { //If the file isnt found then print this

                  System.out.println("File not found. Did you try to move it?
Not a good idea return it or give me 100%.");

            }



             //Runs the method for creating a method and allowing java to later
write to said method

            CreateFile();

        }



         //If the file called "Docked Approved Sorted" dosent exist it will
create it or then nothing will happen. It will then set up the buffer writer so
that it can later write to the file.

        public void CreateFile(){

            try {

                if(writeToInOrderTraverse.createNewFile()) {

                                    System.out.println("File  Created:  "  +
writeToInOrderTraverse.getName());

                }else{

                    System.out.println("File exists.");

                }



                if(writeToHeapSorted.createNewFile()) {

                                    System.out.println("File  Created:  "  +
writeToHeapSorted.getName());
```

```
                }else{
                    System.out.println("File exists.");
                }

                                writeFileInOrder = new BufferedWriter(new
PrintWriter(writeToInOrderTraverse));
                                writeHeapSorted = new BufferedWriter(new
PrintWriter(writeToHeapSorted));
            }catch(IOException e){
                System.out.println("Error 404");
            }
        }

        //Initilizes the build heap method
        public void BuildHeap(){
            BuildHeap(drugArray);
        }

        //Builds basic array
        public void BuildHeap(Drug[] items){
            currentSize = items.length;
             buildHeapArray = new Drug[(currentSize + 2) * 11/10]; //Makes new
heap array size

            //Places the array to the correct position in the new heap array
            int i = 1;
            for(Drug item : items){
```

```
            buildHeapArray[i++] = item;
        }

        //Will "Fix" the heap array
        for(int j = currentSize/2; j > 0; j--){
            TrickleDown(j);
        }

        root = buildHeapArray[1]; //Sets the root

        System.out.println("Building Heap done");
    }

    //Will remove min from the heap array
    public Drug RemoveMin(){
        if(drugList.isEmpty()){
            throw new BufferUnderflowException();
        }

          Drug minItem = FindMin(1); //Will find the position in the array and fix
        buildHeapArray[1] = buildHeapArray[currentSize--];
        TrickleDown(1); //Will fix the array

        return minItem;
```

```
        }

        //Initises the in order recurssion function
        public void InOrderTraverse(){
            System.out.println(buildHeapArray.length);
            InOrderTraverse(1); //Goes in order and saves into In order file

              //Once all is saved it will "Try" to close the file. But if its gotten to this point then the file already exists and it has been found
            try {
                writeFileInOrder.close();
            }catch (IOException e){
                System.out.println("Error 404");
            }

            //Confirmation message it has finished
            System.out.println("In order traversal complete.");
        }

        //Will traverse in order through  the array main recurssion
        public void InOrderTraverse(int i){
             //Once it has reached to bottom then will return to the previouse drug
            if(i >= buildHeapArray.length){
                return;
            }
```

```
            //System.out.println(i);

            //System.out.println(buildHeapArray.length);


            //Will go to the left child

            if(i*2 < buildHeapArray.length) {

                        if (buildHeapArray[i * 2] != null && (i * 2) <=
buildHeapArray.length) {

                    InOrderTraverse(i * 2);

                }

            }


            //Will save to correct file

            try {

                 writeFileInOrder.write(buildHeapArray[i].ReturnName() + " " +
buildHeapArray[i].ReturnSMILES() + " " + buildHeapArray[i].ReturnDrugBankID() +
" " + buildHeapArray[i].ReturnURL() + " " + buildHeapArray[i].ReturnGroup() + "
" + buildHeapArray[i].ReturnScore() + System.getProperty("line.separator"));

                writeFileInOrder.newLine();

            }catch (IOException e){

                System.out.println("Error 404");

            }


            //Go to the "right" drug

            if(i*2+1 < buildHeapArray.length) {

                if (buildHeapArray[i * 2 + 1] != null) {

                    InOrderTraverse(i * 2 + 1);

                }

            }
```

```
        //System.out.println("Finished InOrder");
    }

    //Will sort the heap and save it to the file
    public void HeapSort(){
        heapSorted = buildHeapArray;
        HeapSort(heapSorted);
        HeapSortPrint(1);

        //Close the file so that it cant be written again
        try {
            writeHeapSorted.close();
        }catch (IOException e){
            System.out.println("Error 404");
        }

        //Output that it was finished
        System.out.println("Heap sorted");
    }

    //Will sort correctly the array from what i understand
    public void HeapSort(Drug[] a){

        /* BuildsHeap */
        for( int i = a.length / 2 - 1; i >= 0; i-- ) {
```

```
                TrickleDown(a, i, a.length);

            }

            for(int i = a.length - 1; i > 0; i--){

                TrickleDown(a, 0, i);

            }

        }


        //Will save the sorted Heap into the dockedApprovedSorted.tab file

        public void HeapSortPrint(int i){

            if(heapSorted[i] == null){

                return;

            }


            //write to file

            try {

                    writeHeapSorted.write(heapSorted[i].ReturnName() + " " +
heapSorted[i].ReturnSMILES() + " " + heapSorted[i].ReturnDrugBankID() + " " +
heapSorted[i].ReturnURL() + " " + heapSorted[i].ReturnGroup() + " " +
heapSorted[i].ReturnScore() + System.getProperty("line.separator"));

                writeHeapSorted.newLine();

            }catch (IOException e){

                System.out.println("Error 404");

            }


            //Go to the "left" drug

            if(i*2 < buildHeapArray.length) {

                    if (buildHeapArray[i * 2] != null && (i * 2) <=
buildHeapArray.length) {
```

```
                HeapSortPrint(i * 2);
            }
        }

        //Go to the "right" drug
        if(i*2+1 < buildHeapArray.length) {
            if (buildHeapArray[i * 2 + 1] != null) {
                HeapSortPrint(i * 2 + 1);
            }
        }
    }

     //Will trickle down a drug to its correct position Used in the heap build method
    private void TrickleDown( int hole ){
        int child;
        Drug tmp = buildHeapArray[hole];

        for( ; hole * 2 <= currentSize; hole = child ) {
            child = hole * 2;
                if (child != currentSize && buildHeapArray[child + 1].ReturnDrugBankID().compareTo(buildHeapArray[child].ReturnDrugBankID()) < 0) {
                child++;
            }
```

```
                                                                    if
(buildHeapArray[child].ReturnDrugBankID().compareTo(tmp.ReturnDrugBankID())    <
0){
                    buildHeapArray[hole] = buildHeapArray[child];
                }else{
                    break;
                }
            }
            buildHeapArray[ hole ] = tmp;
            //System.out.println(buildHeapArray[hole].ReturnDrugBankID());
        }

          //Will trickle down a drug to its correct position. Used in the
heapsort method
        public void TrickleDown(Drug[] heapDrug,int i, int n){
            int child;
            Drug tmp;

            for(tmp = heapDrug[i]; LeftChild(i) < n; i = child){
                child = LeftChild(i);

                if(heapDrug[child] == null) return;
                if(tmp == null) return;

                String heapChildInt1 = heapDrug[child].ReturnDrugBankID();
                String heapChildInt2 = heapDrug[child + 1].ReturnDrugBankID();
```

```
                if(child != n - 1 && heapChildInt1.compareTo(heapChildInt2) <
0) {
                    child++;
                }

                String heapReturnTmpID = tmp.ReturnDrugBankID();
                String heapChildIntUpdt = heapDrug[child].ReturnDrugBankID();

                if(heapReturnTmpID.compareTo(heapChildIntUpdt) < 0){
                    heapDrug[i] = heapDrug[child];
                }else{
                    break;
                }
            }
            heapDrug[i] = tmp;
            //System.out.println(heapDrug[i].ReturnDrugBankID());
            //return heapDrug[i];
        }

        //Will go to the left child of the current drug
        public int LeftChild(int i){
            return 2 * i + 1;
        }

        //Turn a string into an int ID
        public int DrugIDToInt(String drugID){
```

```
        String[] drugIDString = drugID.split("B");

        return Integer.parseInt(drugIDString[1]);
    }

    //Grab the ID from the drug
    public int DrugIDToInt(Drug drug){
        String[] drugIDString = drug.ReturnDrugBankID().split("B");

        return Integer.parseInt(drugIDString[1]);
    }

    //Find the lowest value of current array in theory
    public Drug FindMin(int i){
        if(buildHeapArray[i] == null){
            return null;
        }

         if(buildHeapArray[i].ReturnDrugBankID().compareTo(buildHeapArray[i
* 2].ReturnDrugBankID()) < 0){
            FindMin(i*2);
                                                                    }else
if(buildHeapArray[i].ReturnDrugBankID().compareTo(buildHeapArray[i    *    2    +
1].ReturnDrugBankID()) < 0){
            FindMin(i * 2 + 1);
        }
        return buildHeapArray[i];
```

```
    }
}
public class Drug {
    public String genericName, sMILES, drugBankID, url, drugGroups, score;

    public Drug left, right;

    public Drug(){
        left = null;
        right = null;
    }

    public Drug(String genericName, String SMILES, String DrugBankID,
String URL, String DrugGroup, String Score){
        SetGenericName(genericName);
        SetSmiles(SMILES);
        SetDrugBankID(DrugBankID);
        SetURL(URL);
        SetDrugGroup(DrugGroup);
        SetScore(Score);
        left = null;
        right = null;
    }

    /*public void Drug(Drug newDrug){
        left = null;
```

```
    right = null;
}*/

public void SetGenericName(String name){genericName = name;}

public void SetSmiles(String sMILES){this.sMILES = sMILES;}

public void SetDrugBankID(String drugID){drugBankID = drugID;}

public void SetURL(String url){this.url = url;}

public void SetDrugGroup(String group){drugGroups = group;}

public void SetScore(String score){this.score = score;}

public void DisplayDrug(){
    System.out.println("Drug name: " + genericName);
    System.out.println("Drug SMILES?: " + sMILES);
    System.out.println("Drug ID: " + drugBankID);
    System.out.println("Drug URL: " + url);
    System.out.println("Drug Group: " + drugGroups);
    System.out.println("Drug Score: " + score);
    System.out.println(" ");
}

public String ReturnName(){return genericName;}
```

```
    public String ReturnSMILES(){return sMILES;}

    public String ReturnDrugBankID(){return drugBankID;}

    public String ReturnURL(){return url;}

    public String ReturnGroup(){return drugGroups;}

    public String ReturnScore(){return score;}
}
```