OOP Assignment1

Name: Ritabroto Ganguly
Roll: 001910501090 BCSE-II

1. Design a STUDENT class to store roll, name, course, admission date and marks in 5 subjects taken from user. Create an array of STUDENT objects. Provide methods corresponding to admission date and receiving marks, preparing mark sheet. Support must be there to show the number of students who have taken admission.

Inherit Student class and override the input method so as to input the department of each student. Search and display a **sorted list** of students of **one department** or **students** based on scoring criteria. Create an arraylist of students and remove a student based on certain criterion and then call **gc()** and check for free memory.

```
import java.util.*;

import java.util.regex.Pattern;


class STUDENT {

    protected String roll = null; //structured as defined in the description; for this version it has only year (first 2 digits) and count.

    private String name = null;

    private String course = null;

    private Date admission = null; //Admission date

    static final int NUM_SUBJECTS = 5;

    private float marks[] = null; //Marks


    static final int STUDENT_ARRAY_SIZE = 1024;

    private static STUDENT students[] = new STUDENT[STUDENT_ARRAY_SIZE];

    static STUDENT[] getStudents(){STUDENT x[] = students.clone(); return x;}

    private static int studentsAdmitted = 0;

    static int numStudentsAdmitted() {return studentsAdmitted;}
```

```
    static STUDENT search(String r) {
        for(int i = 0; i < studentsAdmitted; i++) if(students[i].roll.equals(r)) return students[i];
        return null; //if not found
    }
    static boolean remove(String r) {//Remove student with given roll
        int i;
        for(i = 0; i < studentsAdmitted; i++) if(students[i].roll.equals(r)) break;
        if(i >= studentsAdmitted) return false;
        for(; i < studentsAdmitted-1; i++) students[i] = students[i+1];
        students[i] = null; //Allow garbage collection
        studentsAdmitted--; return true;
    }

    void input() {//True when valid object has been entered
        System.out.print("Enter name: "); P1.in.skip(Pattern.compile("\\s*")); name =
P1.in.nextLine();
        System.out.print("Enter course: "); course = P1.in.nextLine();
    }

    String getRoll() {return roll;} //may be null if not admitted yet
    String getName() {return name;}
    String getCourse() {return course;}
    Date getAdmissionDate() {return admission;}
    boolean isAdmitted() {return roll != null && admission != null;} //Check if admitted?
    boolean isEvaluated() {return isAdmitted() && marks != null;} //Check if evaluated (marks
given)
    //Get marks of subject subNum(1..5). return -1 if not evaluated.
    float getMarksOfSubject(int subNum) {return isEvaluated() ? -1 : marks[subNum-1];}
    float getTotalMarks() {
```

```
    if(marks == null) return -1;
    float sum = 0; for(float m : marks) sum += m; return sum;
}

void admit() {//Admit student in
    admission = new Date();
    roll = String.format("%ty_%04d", admission, studentsAdmitted + 1);
    students[studentsAdmitted] = this; studentsAdmitted++;
}
void evaluate() {//Give marks
    System.out.print("Enter marks of "+NUM_SUBJECTS+" subjects: ");
    marks = new float[NUM_SUBJECTS];
    for(int i = 0; i < NUM_SUBJECTS; i++) marks[i] = P1.in.nextFloat();
}

boolean printMarkSheet() {
    System.out.println("Student roll: "+roll);
    System.out.println("Name: "+name);
    System.out.println("Course: "+course);
    if(!isAdmitted()) {System.err.println("Student not admitted!"); return false;}
    System.out.printf("Admission date: %td %tB %tY\n", admission, admission, admission);
    if(!isEvaluated()) {System.err.println("Student not evaluated yet."); return false;}
    System.out.print("Marks of "+NUM_SUBJECTS+" subjects: ");
    for(int i = 0; i < NUM_SUBJECTS; i++) System.out.printf("%5.2f ", marks[i]);
    System.out.println(); return true;
}

static void showAllStudents(){
```

```
      for(STUDENT s : students)
        if(s!=null){
          s.printMarkSheet();
          System.out.println("------------------------------");
        }
    }

    public String toString() {return "("+roll+")"+name;}
}

class DepartmentalStudent extends STUDENT {
    //number of students in each department
    private static final HashMap<String, Short> numStudents = new HashMap<String, Short>();
    static short numStudents(String dept_code)
        {return (numStudents.containsKey(dept_code)) ? numStudents.get(dept_code) : 0;}
    static void printAllDeptNumbers() {//Print number of students admitted in each department
        if(numStudents.isEmpty()) return;
        for(String dept : numStudents.keySet()) System.out.println(dept+": "+numStudents.get(dept)+"
students");
    }
    private String dept; //department code
    String getDept() {return dept;}
    @Override void input() {
        super.input();
        System.out.print("Enter department code: ");
        dept = String.format("%4s", P1.in.next()).replace(' ', '0');
    }
    @Override void admit() {
```

```
        super.admit();
        short count = numStudents(dept);
        roll = String.format("%4s_%ty_%04d", dept, super.getAdmissionDate(), count+1);
        numStudents.put(dept, (short)(count+1)); //manual boxing is necessary
    }
    /*@Override boolean printMarkSheet() {
        System.out.printf("Student Department Code: %04s\n", dept);
        return super.printMarkSheet();
    }*/
    static final Comparator<DepartmentalStudent> RANKED_ONMARKS = new
Comparator<DepartmentalStudent>() {
        public int compare(DepartmentalStudent a, DepartmentalStudent b)
            {return Float.compare(b.getTotalMarks(), a.getTotalMarks());}
    };
}


public class P1 {
    static final Scanner in = new Scanner(System.in);
    public static void main(String args[]) {
        System.out.println("1. Add Student");
        System.out.println("2. Assign marks to student (evaluate)");
        System.out.println("3. Print Mark Sheet of Student");
        System.out.println("4. How many students have taken admission?");
        System.out.println("5. Get Ranked List by Department (on total/average marks)");
        System.out.println("6. Remove student");
        System.out.println("7. Display all Students");
        System.out.println("0. Exit");
        int choice; STUDENT s; char choice2; String roll, dept;
```

```
Runtime r = Runtime.getRuntime(); long mem;
ArrayList<DepartmentalStudent> temp;
do {
   System.out.print("Enter choice: "); choice = in.nextInt();
   switch(choice) {
   case 1:
      System.out.print("Is Departmental Student? [y/n] "); choice2 =
Character.toLowerCase(in.next().charAt(0));
      s = (choice2 == 'y') ? new DepartmentalStudent() : new STUDENT();
      s.input();
      s.admit(); System.out.println("Student Admitted: "+s);
      break;
   case 2:
      System.out.print("Enter Roll Number: "); roll = in.next();
      if((s = STUDENT.search(roll)) == null) System.err.println("Student Not Found");
      else {System.out.print(s+": "); s.evaluate();}
      break;
   case 3:
      System.out.print("Enter Roll Number: "); roll = in.next();
      if((s = STUDENT.search(roll)) == null) System.err.println("Student Not Found");
      else s.printMarkSheet();
      break;
   case 4:
      DepartmentalStudent.printAllDeptNumbers();
      System.out.println("In total, "+STUDENT.numStudentsAdmitted()+" students have taken
admission."); break;
   case 5:
      System.out.print("Enter Department Code: "); dept = in.next();
      if(DepartmentalStudent.numStudents(dept) == 0) {
```

```
                System.out.println("No student in this department.");
                break;
            }
            temp = new ArrayList<DepartmentalStudent>();
temp.ensureCapacity(DepartmentalStudent.numStudents(dept));
            STUDENT[] x = STUDENT.getStudents();
            for(int i = 0; i < STUDENT.numStudentsAdmitted(); i++)
                if(x[i] instanceof DepartmentalStudent) {
                    DepartmentalStudent ds = (DepartmentalStudent) x[i];
                    if(ds.getDept().equals(dept)) temp.add(ds);
                }
            Collections.sort(temp, DepartmentalStudent.RANKED_ONMARKS);
            System.out.println("Marks\tStudent");
            for(DepartmentalStudent ds : temp) {
                System.out.printf("%.2f\t%s", ds.getTotalMarks(), ds.toString());
                if(!ds.isEvaluated()) System.out.println("\t(Not Evaluated)");
                else System.out.println();
            }
            break;
        case 6:
            System.out.print("Enter Roll Number: "); roll = in.next();
            if((s = STUDENT.search(roll)) == null) {System.err.println("Student Not Found");
break;}
            System.out.print("Deleting "+s+" ... ");
            STUDENT.remove(roll);
            r.gc(); mem = r.freeMemory();
            s = null;
            r.gc(); mem = r.freeMemory() - mem;
            System.out.println("Memory freed: "+mem+" bytes");
```

```
                break;
            case 7: STUDENT.showAllStudents();break;
            case 0: return;
            default: System.err.print("Invalid Option! ");
           }
        } while (choice != 0);
    }
}
```

```
1. Add Student
2. Assign marks to student (evaluate)
3. Print Mark Sheet of Student
4. How many students have taken admission?
5. Get Ranked List by Department (on total/average marks)
6. Remove student
7. Display all Students
0. Exit
Enter choice: 1
Is Departmental Student? [y/n] y
Enter name: Rg
Enter course: dept
Enter department code: chem
Student Admitted: (chem_21_0001)Rg
Enter choice: 1
Is Departmental Student? [y/n] n
Enter name: Ag
Enter course: dept
Student Admitted: (21_0002)Ag
Enter choice: 2
Enter Roll Number: 21_0002
(21_0002)Ag: Enter marks of 5 subjects: 8 9 7 6 5
Enter choice: 3
Enter Roll Number: 21_0002
Student roll: 21_0002
Name: Ag
Course: dept
Admission date: 05 April 2021
Marks of 5 subjects:  8.00  9.00  7.00  6.00  5.00
Enter choice: 4
chem: 1 students
In total, 2 students have taken admission.
Enter choice: 5
Enter Department Code: dept
No student in this department.

Enter choice: 5
Enter Department Code: chem
Marks   Student
-1.00   (chem_21_0001)Rg        (Not Evaluated)
Enter choice: 6
Enter Roll Number: 21_0002
Deleting (21_0002)Ag ... Memory freed: -3145488 bytes
Enter choice: 7
Student roll: chem_21_0001
Name: Rg
Course: dept
Admission date: 05 April 2021
Student not evaluated yet.
------------------------------
Enter choice: 0
```

2. Design a system for the following scenario:
1. An item list contains item code, name, rate, and quantity for several items.

2. Whenever a new item is added in the list uniqueness of item code is to be checked. Register a new product with its price.

3. Time to time rate of the items may change.
4. Whenever an item is issued or received existence of the item is checked and quantity is updated.
5. In case of issue, availability of quantity is also to be checked.
6. User may also like to know price/quantity available for an item.
7. Find how many items cost more than a given amount. The amount will be a parameter. 8. Remember

that the methods have to return an error code if for example an invalid item code is given **NOTE:**
● The system should be maintained by two types of user, one is Stock entry operator(SEO) and other is

Shopkeeper (SK) and SEO will be the first operator in default case.
● The SEO primarily maintain first 3 operations but SEO users can also maintain all operations (1 to 8)
● SK users can only operates on 4 to 8.
● System should be used for a specific shop type. Ex. Electronics, Book, Grocer etc.. You can design your

system for any one.
● Item Code should be auto generated that includes item name and entry order(1,2,3...)

```
import java.io.*;

import java.util.*;

import classes.FastScanner;


class Item{

  private String code;

  private String name;

  private float rate;
```

```
private int qty;
private String genCode(int qty){
  String x = String.valueOf(qty);
  if(qty<10)
    x = "00"+qty;
  else if(qty<100)
    x = "0"+qty;

  return x;
}
public Item(String name,float rate,int qty){
  this.code = (name.toUpperCase() + genCode(qty));
  this.name = name;
  this.rate = rate;
  this.qty = qty;
}
public void changeRate(float newRate){rate = newRate;}
public void changeQty(int newQty){qty = newQty;}
public int getQty(){return qty;}
public float getPrice(){return rate;}
public String getCode(){return code;}
public String getName(){return name;}
public String toString(){return (name+" "+qty+" "+rate+" "+code);}
public boolean equals(Object o){
 if(o==this) return true;
 if(!(o instanceof Item)) return false;
 Item x = (Item)o;
 return x.getCode().equals(code);
```

```
  }
  public int hashCode(){return -1;}
}

class ItemList{
  private static FastScanner s = new FastScanner();
  private ArrayList<Item> il;
  private int desig = 1;//default designation for SEO
  private Item check(String code){//5
    for(Item i : il)
      if(i.getCode().equals(code))
        return i;

    return null;
  }

  private ArrayList<Item> checkName(String name){
    ArrayList<Item> it = new ArrayList<Item>();
    for(Item i : il)
      if(i.getName().equals(name))
        it.add(i);

    return it;
  }

  private ItemList(int items){
    il = new ArrayList<Item>();
    for(int i=0;i<items;i++){
```

```
      String name="";float rate=-1;int qty=-1;
     while(qty==-1){
      try{
       name = s.next();rate = s.nextFloat();qty = s.nextInt();
      }catch(Exception e){System.out.println("Invalid Input(s), Try Again.");qty=-1;}
     }
      il.add(new Item(name,rate,qty));
   }
   System.out.println("Signed in as SEO");
  }
  private ItemList(Item[] i){il = new
ArrayList<Item>();Collections.addAll(il,i);System.out.println("Signed in as SEO");}


  public static ItemList getInstance(int desig,int items){
   if(desig!=1){System.out.println("Unauthorised! Have to be SEO to access this method");return
null;}
   ItemList i = new ItemList(items);return i;
  }
  public static ItemList getInstance(int desig,Item[] items){
   if(desig!=1){System.out.println("Unauthorised! Have to be SEO to access this method");return
null;}
   ItemList i = new ItemList(items);return i;
  }


  public void changeDesig(){
   if(desig!=1){desig = 1;System.out.println("Signed in as SEO");}
   else{desig = 2;System.out.println("Signed in as ShopKeeper");}
  }
```

```
 public static int input(String st){
  int x=-1;
  while(x==-1){
   System.out.print(st);
   try{
    x = s.nextInt();
   }catch(Exception e){System.out.println("Invalid Input, Try Again.");x=-1;}
  }
  return x;
}

 public void addNewItem(){//2
  if(desig!=1){System.out.println("Unauthorised! Have to be SEO to access this method");return;}
  int x = 1;
  while(x!=0){
   System.out.print("Enter Item Name,Rate,Quantity: ");
   Item it = null;
   try{
    it = new Item(s.next(),s.nextFloat(),s.nextInt());
   }catch(Exception e){System.out.println("Invalid Input(s)");return;}
   if(il.contains(it))System.out.println("Item Already exists");
   else {il.add(it);System.out.println("Item Added");}
   x = input("Add more items (yes=1,no=0): ");
  }
 }

public void updateStock(String code){//4
  Item i = check(code);
```

```
  if(i!=null){
    int x = input("Issued or received? (issued=1,received=2): ");
    if(x==1){
      x = input("How many issued: ");
      if(x>i.getQty())System.out.println("Insufficient stock");
      else i.changeQty((i.getQty()-x));
    }
    else{
      x = input("How many received: ");
      i.changeQty((i.getQty()+x));
    }
  }else
    System.out.println("No item found with itemcode "+code);
}

public void changeRate(String code){//3
  if(desig!=1){System.out.println("Unauthorised! Have to be SEO to access this method");return;}
  Item i = check(code);
  if(i!=null){
    float x=0;
    while(x==0){
     System.out.print("Enter New Rate: ");
     try{
      x = s.nextFloat();
     }catch(Exception e){System.out.println("Invalid Input, Try Again.");x=0;}
    }
    i.changeRate(x);
  }else
```

```
    System.out.println("No item found with itemcode "+code);
}

public void getDetailsByName(String name){//6
  ArrayList<Item> i = checkName(name);
  if(i.size()<=0) System.out.println("No item exists with name "+name);
  else for(Item it : i) System.out.println(it.toString());
}
public void getDetails(String code){//6
  Item i = check(code);
  if(i==null) System.out.println("No item exists with code "+code);
  else System.out.println(i.toString());
}

public void itemsMoreThan(float amount){//7
 boolean e = false;
 for(Item i : il)
    if(i.getPrice()>amount){
      System.out.println(i.toString());
      e = true;
    }

 if(e==false)
   System.out.println("No item with price less than " + amount);
}

public String toString(){
 StringBuilder x = new StringBuilder("");
```

```
  for(Item i : il){
     x.append(i.toString()).append("\n----------------------------\n");
  }
  return x.toString();
 }
}

class Main{
        public static void main(String[] args) throws IOException{
                FastScanner s = new FastScanner();
            Item[] y = new Item[5];
            for(int i=1;i<6;i++)
                y[i-1] = new Item("Item"+i,(float)(i*5.5),i*3);
            ItemList il = null;int x;
           while(il==null){
            x = ItemList.input("SEO or ShopKeeper (SEO=Any key other than 2,Shopkeeper=2): ");
            if(x!=2)x=1;
            il = ItemList.getInstance(x,y);
           }
            System.out.println(il);
            while(true){
              x = ItemList.input("Choose an option (1=change user,2=add new item,3=change
rate,4=update stock,5=get details,6=items more a given amount,7=display, Escape=8): ");
              if(x==8) break;
              switch(x){
                case 1:
                x = ItemList.input("SEO or ShopKeeper (SEO=Any key other than 2,Shopkeeper=2):
");
                if(x!=2)x=1;
```

```
il.changeDesig();break;
case 2:
il.addNewItem();break;
case 3:
try{System.out.print("Enter code: "); String code = s.next(); il.changeRate(code);}
catch(Exception e){System.out.println("Invalid Input code");}
break;
case 4:
try{System.out.print("Enter code: "); String code = s.next(); il.updateStock(code);}
catch(Exception e){System.out.println("Invalid Input code");}
break;
case 5:
x = ItemList.input("Get details by name or code (name=1,code=2): ");
if(x==2){
 try{System.out.print("Enter code: "); String code = s.next(); il.getDetails(code);}
 catch(Exception e){System.out.println("Invalid Input code");}
}else{
 try{System.out.print("Enter name: "); String name = s.next();
il.getDetailsByName(name);}
 catch(Exception e){System.out.println("Invalid Input name");}
}
break;
case 6:
x = ItemList.input("Enter amount: ");
il.itemsMoreThan(x);
break;
case 7:
System.out.println(il);break;
```

```
                default: System.out.println("Invalid option");
            }
        }
    }
}
```

```
SEO or ShopKeeper (SEO=Any key other than 2,Shopkeeper=2): 7
Signed in as SEO
Item1 3 5.5 ITEM1003
-----------------------------
Item2 6 11.0 ITEM2006
-----------------------------
Item3 9 16.5 ITEM3009
-----------------------------
Item4 12 22.0 ITEM4012
-----------------------------
Item5 15 27.5 ITEM5015
-----------------------------

Choose an option (1=change user,2=add new item,3=change rate,4=update stock,5=get details,6=items more a given amount,7=display, E
scape=8): 2
Enter Item Name,Rate,Quantity: Item1 4 7
Item Added
Add more items (yes=1,no=0): 0
Choose an option (1=change user,2=add new item,3=change rate,4=update stock,5=get details,6=items more a given amount,7=display, E
scape=8): 3
Enter code: ITEM1003
Enter New Rate: 7
Choose an option (1=change user,2=add new item,3=change rate,4=update stock,5=get details,6=items more a given amount,7=display, E
scape=8): 5
Get details by name or code (name=1,code=2): Item1
Invalid Input, Try Again.
Get details by name or code (name=1,code=2): 1
Enter name: Item1
Item1 3 7.0 ITEM1003
Item1 7 4.0 ITEM1007
Choose an option (1=change user,2=add new item,3=change rate,4=update stock,5=get details,6=items more a given amount,7=display, E
scape=8): 6
Enter amount: 11
Item3 9 16.5 ITEM3009
Item4 12 22.0 ITEM4012
Item5 15 27.5 ITEM5015

Choose an option (1=change user,2=add new item,3=change rate,4=update stock,5=get details,6=items more a given amount,7=display, E
scape=8): 7
Item1 3 7.0 ITEM1003
-----------------------------
Item2 6 11.0 ITEM2006
-----------------------------
Item3 9 16.5 ITEM3009
-----------------------------
Item4 12 22.0 ITEM4012
-----------------------------
Item5 15 27.5 ITEM5015
-----------------------------
Item1 7 4.0 ITEM1007
-----------------------------

Choose an option (1=change user,2=add new item,3=change rate,4=update stock,5=get details,6=items more a given amount,7=display, E
scape=8): 8
```

3. Write a program Parentheses.java that reads in a text stream from standard input and uses a ***stack*** to determine whether or not its parentheses are properly balanced. For example, your program should print true for [()]{}{[()()]()} and false for [(]). You need to implement the ***stack*** class by yourself.

Parentheses.java

```
import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.io.IOException;

public class Parentheses {
 private static class Bracket {
    char type;
    int position;

    Bracket(char type, int position) {
        this.type = type;
        this.position = position;
    }

    boolean Match(char c) {
        if (this.type == '[' && c == ']')
            return true;
        if (this.type == '{' && c == '}')
            return true;
        if (this.type == '(' && c == ')')
            return true;
        return false;
```

```
    }
}


    public static void main(String[] args) throws IOException {
        InputStreamReader input_stream = new InputStreamReader(System.in);
        BufferedReader reader = new BufferedReader(input_stream);
        String text = reader.readLine();


        Stack<Bracket> opening_brackets_stack = new Stack<Bracket>();
        for (int position = 0; position < text.length(); ++position) {
            char next = text.charAt(position);
            Bracket bracket = new Bracket(next, position);


            if (!(next == '(' || next == '[' || next == '{' || next == ')' || next == ']' || next == '}')){//if not a
bracket
                continue;
            }


            if (next == '(' || next == '[' || next == '{') {//if opening bracket
                opening_brackets_stack.push(bracket);
            } else {
                if (opening_brackets_stack.empty()){//if closing bracket without opening bracket
                    System.out.println((position+1));
                    return;
                }
                Bracket bracketcheck = opening_brackets_stack.pop();
                if (bracketcheck==null || !bracketcheck.Match(bracket.type)){//if closing bracket
mismatched with opening bracket
                    System.out.println((position+1));
```

```
                return;
            }
        }
    }

    if (opening_brackets_stack.empty()){//if all brackets matched
        System.out.println("No errors");
    }else {//at least 1 bracket left unpaired/unmatched
        System.out.println((opening_brackets_stack.peek().position+1));
    }
  }
}
```

Stack.java

```
import java.io.*;
import java.util.*;

public class Stack<T>{
  private class Node{
    T value;
    Node next;
    Node prev = null;
    Node(T v){value = v; next = null;}
  }

  private Node top = null;

  public void push(T v){
```

```
    Node newNode = new Node(v);
    if(top!=null){top.next = newNode; newNode.prev = top;}
    top = newNode;
  }

  public T pop(){
    if(top==null) return null;
    T x = top.value;
    top = top.prev;
    return x;
  }

  public T peek(){
    return top.value;
  }

  public boolean empty(){return top==null;}
}
```

```
Ritobrotos-MacBook-Air:q3 rgdgr8$ java Parentheses
{{{{{{{{{{{{{]]]]]]]]]]]]]]]]
14
```

4. Create a class diagram and Java code for the following system and scenario, taking into account the possibility of future extensions. "The system is a command line utility that prints a short 'quote of the day' on the user's terminal when run. To begin with the quote is selected randomly from a set of hard-coded strings within the program itself, but that might change later on -- the quotes might be based on the **user's history, the time of day, the date,** etc.. **Scenario**

```
import java.io.*;

import java.util.*;

import java.time.*;

class Quotes{

  private ArrayList<String> quotes;

  private static final Random rand = new Random();

  private LocalDateTime ldt;

  private int[] last = null; //last few shown quotes, so that same quotes are not repeatedly implemented as an circular queue.

  private int lastIndex = -1;

  private int rotateListIndex(int i,int size,int frontIndex){if(i>=(size-1))return frontIndex;return ++i;}

  private void pushLast(int i){ lastIndex = rotateListIndex(lastIndex,last.length,0); last[lastIndex] = i;}


  public Quotes(int lastSize) throws Exception{

   quotes = new ArrayList<>();

   quotes.add("Independence Day");

   quotes.add("Teachers' Day");

   quotes.add("God helps them that help themselves. -- Benjamin Franklin");

   quotes.add("Happiness is not a reward - it is consequence. Suffering is not a punishment - it is a result. -- Robert Green Ingersoll");

   quotes.add("Future. That period of time in which our affairs prosper, our friends are true and our happiness is assured. -- Ambrose Bierce");
```

```
quotes.add("Honesty is the first chapter of the book of wisdom. --Thomas Jefferson");
quotes.add("book of wis--Thomas Jefferson");
quotes.add("--Thomas Jefferson");
quotes.add("--Tho");
quotes.add("--");

last = new int[lastSize];

FileInputStream f = null;
try{
   f = new FileInputStream(new File("lasts.txt"));
}catch(Exception e){System.out.println("Error in opening file");}
FastScanner s = new FastScanner(f);

int i;
for(i=0;i<lastSize;i++) last[i] = quotes.size();
for(i=0;i<lastSize;i++){
    try{
     last[i] = s.nextInt();
    }catch(Exception e){/*System.out.println("Error while reading");e.printStackTrace();*/}
}

try{
   lastIndex = s.nextInt();
   f.close();
}catch(Exception e){/*System.out.println("Error while reading lastIndex");e.printStackTrace();*/}
```

```
}

 public int[] getLasts(){
    int[] x = new int[last.length];
    for(int i=0;i<x.length;i++)
       x[i] = last[i];
    return x;
 }

 public int getLastIndex(){return lastIndex;}

 //public void addQuote(String s){quotes.add(s);}
 public void show(){
   ldt =LocalDateTime.now();
   int month = ldt.getMonthValue();
   int day = ldt.getDayOfMonth();
   //System.out.println("dd/mm = " + day + "/" + month);
   if(month==8 && day==15)
     System.out.println(quotes.get(0));
   else if(month==9 && day==5)
     System.out.println(quotes.get(1));
   else{
     int i = rand.nextInt((quotes.size()-2))+2;
     int[] temp = new int[last.length];
     for(int k=0;k<last.length;k++) temp[k] = last[k];
     Arrays.sort(temp);
     /*for(int l : temp){System.out.print(l+" ");}
```

```
      System.out.println("  i="+i);*/


      for(int j=0;j<temp.length;j++){
        if(i==temp[j]){
          i = rotateListIndex(i,quotes.size(),2);
          if(j==(temp.length-1))
            j=-1;
        }
        else if(i<temp[j]) break;
      }
      pushLast(i);
      /*for(int l : last){System.out.print(l+" ");}
      System.out.println("  i="+i);*/
      System.out.println(quotes.get(i));
    }
  }
}


class QuoteOfTheDay{
        public static void main(String[] args) throws Exception{
          try {
            File myObj = new File("lasts.txt");
            if (myObj.createNewFile()) {
            System.out.println("File created: " + myObj.getName());
            } //else {
            //System.out.println("File already exists.");
            //}
```

```
        } catch (IOException e) {
            System.out.println("An error occurred while creating file");
            e.printStackTrace();
            return;
        }

        Quotes q = new Quotes(2);int x[] = null; int y = -1;
        //while(s.nextInt()!=0)
         q.show();
         try {
             FileWriter myWriter = new FileWriter("lasts.txt");
             x = q.getLasts();
             y = q.getLastIndex();
             for(int i = 0;i<x.length;i++)
                 myWriter.write(x[i]+" ");
             myWriter.write("\n"+y);
             myWriter.close();
             //System.out.println("Successfully wrote to the file.");
         } catch (IOException e) {
             System.out.println("An error occurred while writing into file");
             e.printStackTrace();
         }

         //for(int i=0;i<x.length;i++) System.out.print(x[i] + " ");
         //System.out.println("\n"+y);
    }
}
```

```
Ritobrotos-MacBook-Air:q4 rgdgr8$ java QuoteOfTheDay
--Tho
Ritobrotos-MacBook-Air:q4 rgdgr8$ java QuoteOfTheDay
--Thomas Jefferson
Ritobrotos-MacBook-Air:q4 rgdgr8$ java QuoteOfTheDay
Future. That period of time in which our affairs prosper, our friends are true and our happiness is
assured. -- Ambrose Bierce
Ritobrotos-MacBook-Air:q4 rgdgr8$ java QuoteOfTheDay
Honesty is the first chapter of the book of wisdom. --Thomas Jefferson
Ritobrotos-MacBook-Air:q4 rgdgr8$ java QuoteOfTheDay
--
Ritobrotos-MacBook-Air:q4 rgdgr8$ java QuoteOfTheDay
--Tho
Ritobrotos-MacBook-Air:q4 rgdgr8$ java QuoteOfTheDay
God helps them that help themselves. -- Benjamin Franklin
```

5. Indexing a book. Write a program that reads in a text file from standard input and compiles an alphabetical index of which ***words/phrases*** appear on which lines, as in the following input. Ignore case and punctuation. For each word maintain a list of location on which it appears. Try to use HashTable and/or HashMap class (of java.util).

```
import java.io.*;

import java.util.*;

import classes.FastScanner;


class Main{

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

                FastScanner s = new FastScanner();

            System.out.print("Enter file name: ");

            String fn = s.next();

            File f = new File(fn);

            if(!f.exists() || !f.canRead()){

              System.out.println("Can't work with "+fn);

              System.exit(0);

            }
```

```
BufferedReader br = new BufferedReader(new FileReader(f));
StringTokenizer st = null;
String tt = br.readLine();
int line = 1;
HashMap<String,LinkedHashSet<Integer>> hm = new HashMap<>();

while(tt!=null){
  tt = tt.toLowerCase().replaceAll("\\p{Punct}","");
  st = new StringTokenizer(tt);
  while(st.hasMoreTokens()){
     String x = st.nextToken();
     LinkedHashSet<Integer> lhs = null;
     if(hm.get(x)==null)
       lhs = new LinkedHashSet<>();
     else
       lhs = hm.get(x);
     lhs.add(line);
     hm.put(x,lhs);
  }
  tt = br.readLine();
  line++;
}

for(Map.Entry<String,LinkedHashSet<Integer>> me : hm.entrySet()){
   System.out.print(me.getKey()+": ");
   for(Integer i : me.getValue())
     System.out.print(i+" ");
   System.out.println("");
```

```
        }
    }
}
```

```
1 Hi, my name is
2 Ritabroto Ganguly
3 JU CSE
4 2nd Year
```

```
Ritobrotos-MacBook-Air:q5 rgdgr8$ java Main
Enter file name: book.txt
hi: 1
cse: 3
year: 4
name: 1
2nd: 4
ganguly: 2
is: 1
ju: 3
my: 1
ritabroto: 2
```

6. Design and create a hospital information system with the following scenarios.
● Register a new patient.
● Each patient is assigned to one doctor, but a doctor can have any number of patients. Patients check in to the

hospital and assigned a doctor if they don't already have one.
● While in the hospital, doctors record various observations about each patient at various times. Examples of

observations are blood pressure and temperature. Record test results for a patient.
● The hospital keeps track of all the observations for a given patient until they check out of the hospital. Obtain all of a patient’s information given the social security number.

```
import java.io.*;
```

```
import java.util.*;
import classes.FastScanner;

class Hospital{
  private class Patient{
    private String id;
    Patient(String ssn){id = ssn;}
    public String getPatientId(){return id;}
    public String toString(){return id;}
    public boolean equals(Object o){
      if(o==this)
        return true;

      if(o==null)
        return false;

      if(o instanceof Patient){
        Patient x = (Patient)o;
        return (x.getPatientId().equals(id));
      }

      return false;
    }
    public int hashCode(){return -1;}
  }

  private static int dids = 1;
  private class Doctor{
```

```
    private String name;
    private ArrayList<Patient> pa;
    Doctor(){pa = new ArrayList<>();name = "Doctor"+dids;dids++;}
    public void addPatient(Patient p){pa.add(p);}
    public void showAllReports(){
      for(Patient p : pa){
         System.out.println("Report for:");
         System.out.println(p);
      }
    }
    public boolean hasPatient(Patient p){
      int x = pa.indexOf(p);
      return (x==-1 ? false : true);
    }
    public void showPatientReport(Patient p){
      System.out.println("Report for:\n"+p);
    }
    public int getNumOfPatients(){return pa.size();}
    public String toString(){return name;}
  }


  private ArrayList<Doctor> da;
  public Hospital(int doctors){da = new ArrayList<>(doctors);for(int i=0;i<doctors;i++) da.add(new
Doctor());}
  private void assignDoctor(Patient p){
    int min = Integer.MAX_VALUE; int mini = -1;
    for(int i=0;i<da.size();i++){
       if(da.get(i).getNumOfPatients()<min){
```

```
            min = da.get(i).getNumOfPatients();
            mini = i;
          }
      }
      da.get(mini).addPatient(p);
    }
    public void registerPatient(String ssn){Patient p = new Patient(ssn);assignDoctor(p);}
    public void showAllRecords(){
      for(Doctor d : da){
        System.out.println("Under doctor "+d+": ");
        d.showAllReports();
      }
      System.out.println("-----------------------------------");
    }
    public void showPatientReport(String ssn){
       Patient p = new Patient(ssn);
       for(Doctor d : da)
          if(d.hasPatient(p)){
            d.showPatientReport(p);
            return;
          }
       System.out.println("Patient not registered!");
    }
}

class Main{
        public static void main(String[] args) throws IOException{
            Hospital h = new Hospital(5);
```

```
        for(int i=0;i<10;i++)
        h.registerPatient("p"+i);

        h.showAllRecords();
        h.showPatientReport("p11");
      }
  }
```

```
Ritobrotos-MacBook-Air:q6 rgdgr8$ java Main
Under doctor Doctor1:
Report for:
p0
Report for:
p5
Under doctor Doctor2:
Report for:
p1
Report for:
p6
Under doctor Doctor3:
Report for:
p2
Report for:
p7
Under doctor Doctor4:
Report for:
p3
Report for:
p8
Under doctor Doctor5:
Report for:
p4
Report for:
p9
-----------------------------------
Patient not registered!
```