import java.io.BufferedReader;

import java.io.FileNotFoundException;

import java.io.FileReader;

import java.io.IOException;

import java.util.Scanner;

//trabajo ralizado por Benjamin Pastor

public class App {

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

        Scanner teclado = new Scanner(System.in);

        int grado, j = 0;

        TDAalumnos[] egresados = new TDAalumnos[30];

        // las siguientes lineas realizan la carga de los alumnos

        // en sus grados

        TDAalumnos[][] grados = new TDAalumnos[7][30];

        cargarAlumnos(grados);

        // carga de reprobados (los llame repitentes)

        int cantRepitentes = cantRepitentes();

        int[] repitentes = new int[cantRepitentes];

        cargarRepitentes(repitentes);

        // carga de ingresantes

        int cantIngresos = cantIngresantes();

        TDAalumnos[] ingresantes = new TDAalumnos[cantIngresos];

        cargarIngresantes(ingresantes);

        mostrarMatriz(grados);

        int i = 0, ocupado = 0;

        nuevoAlumno(grados);

        mostrarMatriz(grados);

    }

    // este modulo muestra la matriz cargada de los alumnos en sus grados

    // respectivos

    public static void mostrarMatriz(TDAalumnos[][] grados) {

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

            System.out.print("grado" + (i + 1) + ": ");

            for (int j = 0; j < grados[0].length; j++) {

                TDAalumnos alumno = grados[i][j];

                if (grados[i][j] != null) {

                    System.out.print(alumno.getLegajo() + ",");

                } else {

                    System.out.print("vacio, ");

                }

            }

            System.out.println("");

        }

    }

    // este modulo genera unAlumno y retorna para cargarlo despues

    public static TDAalumnos crearAlumno(String alumno) {

        TDAalumnos unAlumno = new TDAalumnos("", "", 0, 0, 0);

        String[] armarAlumno = alumno.split(";");

        if (armarAlumno.length == 5) {

            String apellido = armarAlumno[0];

            String nombre = armarAlumno[1];

            int legajo = Integer.parseInt(armarAlumno[2]);

            int grado = Integer.parseInt(armarAlumno[3]);

            double promedio = Double.parseDouble(armarAlumno[4]);

            unAlumno = new TDAalumnos(apellido, nombre, legajo, grado, promedio);

        }

        return unAlumno;

    }

    // este modulo realiza la carga en la matriz del alumno creado con "crearAlumno"

    public static void cargarAlumnos(TDAalumnos[][] grados) {

        String ListaIngresantes = "C:\\Users\\Benjamin\\Desktop\\ListaIngresantes.txt";

        String linea;

        TDAalumnos unalumno;

        try {

            FileReader lectorArchivo = new FileReader(ListaIngresantes);

            BufferedReader bufferLector = new BufferedReader(lectorArchivo);

            while ((linea = bufferLector.readLine()) != null) {

                unalumno = crearAlumno(linea);

                if (unalumno.getGrado() >= 1 && unalumno.getGrado() <= 7) {

                    int grado = unalumno.getGrado() - 1;

                    boolean alumnoCargado = false;

                    for (int j = 0; j < 30 && alumnoCargado == false; j++) {

                        if (grados[grado][j] == null) {

                            grados[grado][j] = unalumno;

                            alumnoCargado = true;

                        }

                    }

                }

            }

            bufferLector.close();

        } catch (FileNotFoundException ex) {

            System.err.println(ex.getMessage() + "No se encontro el archivo");

        } catch (IOException ex) {

            System.err.println("Error en lectura o escritura");

        }

    }

    // este modulo calcula la cantidad de repitentes (reprobados)

    // para asi poder definir un arreglo de repitentes de la longitud necesaria

    public static int cantRepitentes() {

        int contador = 0;

        String linea;

        String ListaDesaprobados = "C:\\Users\\Benjamin\\Desktop\\ListaDesaprobados.txt";

        try {

            FileReader lectorArchivo = new FileReader(ListaDesaprobados);

            BufferedReader bufferLector = new BufferedReader(lectorArchivo);

            while ((linea = bufferLector.readLine()) != null) {

                contador++;

            }

            bufferLector.close();

        } catch (FileNotFoundException ex) {

            System.err.println(ex.getMessage() + "No existe el archivo.");

        } catch (IOException ex) {

            System.err.println("Error escribiendo o leyendo el archivo.");

        }

        return contador;

    }

    // este modulo hace la carga de los legajos de los repitentes a su arreglo

    public static void cargarRepitentes(int[] repitentes) {

        int i = 0, legajo;

        String linea;

        String ListaDesaprobados = "C:\\Users\\Benjamin\\Desktop\\ListaDesaprobados.txt";

        try {

            FileReader lectorArchivo = new FileReader(ListaDesaprobados);

            BufferedReader bufferLector = new BufferedReader(lectorArchivo);

            while ((linea = bufferLector.readLine()) != null && i < repitentes.length) {

                legajo = Integer.parseInt(linea);

                repitentes[i] = legajo;

                i++;

            }

            bufferLector.close();

        } catch (FileNotFoundException ex) {

            System.err.println(ex.getMessage() + "No existe el archivo.");

        } catch (IOException ex) {

            System.err.println("Error escribiendo o leyendo el archivo.");

        }

    }

    // este modulo calcula la cantidad de ingresantes para definir el largo del

    // arreglo

    // de ingresantes

    public static int cantIngresantes() {

        int contador = 0;

        String linea;

        String ListaIngresantes = "C:\\Users\\Benjamin\\Desktop\\ListaIngresantes.txt";

        try {

            FileReader lectorArchivo = new FileReader(ListaIngresantes);

            BufferedReader bufferLector = new BufferedReader(lectorArchivo);

            while ((linea = bufferLector.readLine()) != null) {

                contador++;

            }

            bufferLector.close();

        } catch (FileNotFoundException ex) {

            System.err.println(ex.getMessage() + "No existe el archivo.");

        } catch (IOException ex) {

            System.err.println("Error escribiendo o leyendo el archivo.");

        }

        return contador;

    }

    // este modulo realiza la carga de los ingresantes

    public static void cargarIngresantes(TDAalumnos[] ingresantes) {

        String ListaIngresantes = "C:\\Users\\Benjamin\\Desktop\\ListaIngresantes.txt";

        String linea;

        TDAalumnos unalumno;

        int i = 0;

        try {

            FileReader lectorArchivo = new FileReader(ListaIngresantes);

            BufferedReader bufferLector = new BufferedReader(lectorArchivo);

            while ((linea = bufferLector.readLine()) != null) {

                unalumno = crearAlumno(linea);

                boolean alumnoCargado = false;

                for (int j = 0; j < 30 && alumnoCargado == false; j++) {

                    if (ingresantes[i] == null) {

                        ingresantes[i] = unalumno;

                        alumnoCargado = true;

                    }

                }

            }

            bufferLector.close();

        } catch (FileNotFoundException ex) {

            System.err.println(ex.getMessage() + "No se encontro el archivo");

        } catch (IOException ex) {

            System.err.println("Error en lectura o escritura");

        }

    }

    // este modulo calcula de forma recursiva el promedio de cada grado

    public static double calcularPromedio(TDAalumnos[][] grados, int i, int j) {

        double prom = 0;

        TDAalumnos alumno;

        if (j == grados[i].length - 1 || grados[i][j + 1] == null) {

            alumno = grados[i][j];

            prom = alumno.getPromedio();

        } else {

            alumno = grados[i][j];

            prom = alumno.getPromedio() + calcularPromedio(grados, i, j + 1);

        }

        return prom;

    }

    // este modulo cuenta la cantidad de vacantes

    public static int cantidadLugares(TDAalumnos[][] grados, int i, int j, int ocupados) {

        int vacantes = ocupados, suma = 0;

        if (i < grados.length) {

            if (j < grados[0].length && grados[i][j] != null) {

                vacantes = cantidadLugares(grados, i, j + 1, ocupados + 1);

            } else {

                vacantes = cantidadLugares(grados, i + 1, 0, ocupados);

            }

        }

        return grados.length \* grados[0].length - vacantes;

    }

    // este modulo realiza el alta del nuevo alumno pidiendo los datos al usuario

    public static void nuevoAlumno(TDAalumnos[][] grados) {

        Scanner teclado = new Scanner(System.in);

        int i, j = 0;

        boolean ingresado = false;

        String alumno, nombre, apellido, legajo, grado, promedio;

        System.out.print("ingrese el apellido: ");

        apellido = teclado.nextLine();

        System.out.print("Ingrese el nombre: ");

        nombre = teclado.nextLine();

        System.out.print("ingresse el legajo: ");

        legajo = teclado.nextLine();

        System.out.print("Ingrese el grado: ");

        grado = teclado.nextLine();

        System.out.print("Ingrese el  promedio: ");

        promedio = teclado.nextLine();

        alumno = apellido + ";" + nombre + ";" + legajo + ";" + grado + ";" + promedio;

        i = Integer.parseInt(grado) - 1;

        TDAalumnos unAlumno = crearAlumno(alumno);

        while (i <= grados.length - 1 && j < grados[0].length && ingresado == false) {

            if (grados[i][j] == null) {

                grados[i][j] = unAlumno;

                ingresado = true;

                System.out.println("El alumno se cargo correctamente");

            } else {

                j++;

            }

        }

    }

}

Tda:

public class TDAalumnos {

    private String apellido;

    private String nombre;

    private int legajo;

    private int grado;

    private double promedio;

    // constructor

    public TDAalumnos(String a, String n, int l, int g, double p) {

        this.apellido = a;

        this.nombre = n;

        this.legajo = l;

        this.grado = g;

        this.promedio = p;

    }

    // observadores

    public String getNombre() {

        return nombre;

    }

    public String getApellido() {

        return apellido;

    }

    public int getLegajo() {

        return legajo;

    }

    public int getGrado() {

        return grado;

    }

    public double getPromedio() {

        return promedio;

    }

    // modificadores

    public void setGrado(int g) {

        grado = g;

    }

    // toString re piola

    public String toString() {

        return ("apellido: " + apellido + ", nombre: " + nombre + ", legajo: " + legajo + ", grado: " + grado

                + ", promedio: " + promedio);

    }

}

Por las dudas:

// el siguiente modulo se encarga de realizar el proceso de pasar de grado

    public static void pasarDeGrado(TDAalumnos[][] grados, TDAalumnos[] egresados, int[] repitentes,

            TDAalumnos[] ingresantes) {

        boolean esRepitente = false;

        int i = grados.length - 1;

        TDAalumnos[][] nuevosGrados = new TDAalumnos[7][30];

        // este for recorre las filas

        for (i = grados.length - 1; i == 0; i--) {

            boolean nulo = false;

            int j = 0;

            // while recorre las columnas hasta que se llegue a una posicion nula

            while (j < grados[i].length && nulo == false) {

                if (grados[i][j] != null) {

                    boolean finalizar = false;

                    // utilizo este for para recorrer el arreglo de repitentes hasta que termine

                    // o encuentre algun repitente

                    for (int k = 0; k < repitentes.length && finalizar == false; k++) {

                        if (grados[i][j].getLegajo() == repitentes[k]) {

                            esRepitente = true;

                            finalizar = true;

                        }

                    }

                    // en este if si el alumno no es repitente y esta en la ultima fila

                    // lo cargo al arreglo de egresados

                    if (esRepitente == false && i == grados.length - 1) {

                        egresados[j] = grados[i][j];

                        grados[i][j] = null;

                    }

                    // si el alumno no estaba en la ultima fila lo cargo a la siguiente fila que le

                    // corresponda

                    if (esRepitente == false && i < grados.length - 1) {

                        int cont = 0;

                        TDAalumnos unAlumno = grados[i][j];

                        unAlumno.setGrado(unAlumno.getGrado() + 1);

                        nuevosGrados[unAlumno.getGrado()][j] = unAlumno;

                        cont++;

                    }

                } else {

                    nulo = true;

                }

            }

        }

    }