Ejemplo uso de HashMap caso promedio notas.

El profesor de español desea gestionar la información de notas obtenidas por sus estudiantes, para eso necesita una aplicación qué calcule el promedio de 3 notas parciales de la materia por cada estudiante para calcular la nota final (promedio de las 3 notas) y así determinar cuantos estudiantes ganan y cuantos estudiantes pierden, un estudiante gana cuando el promedio de las 3 notas es mayor o igual a 3.5.

Haga un algoritmo qué solicite el nombre, documento, materia y las 3 notas parciales para n estudiantes, el sistema deberá permitir el ingreso de las notas y cuando finalice deberá imprimir:

1. Cantidad total de estudiantes validados
2. Cantidad total de notas ingresadas (si son 3 estudiantes entonces serian 9 notas ingresadas en total)
3. Sumatoria total de las notas ingresadas (suma de las notas del punto anterior)
4. Promedio de las notas finales obtenidas. (promedio de las notas finales de cada estudiante.)
5. Cantidad de estudiantes qué ganan la materia (ganan cuando su promedio es mayor o igual a 3.5)
6. Cantidad de estudiantes qué pierden la materia (pierden cuando su promedio es menor a 3.5)
7. Cantidad de estudiantes qué pueden recuperar (cuando su nota final es superior a 2)
8. Cantidad de estudiantes qué pierden sin recuperación.

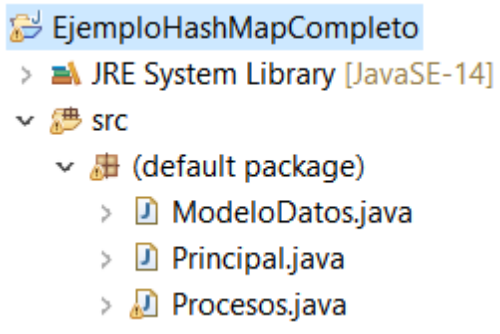
Solución

Para solucionar este ejercicio vamos a hacer uso de una estructura de datos de tipo HashMap principal que almacenará los datos de los estudiantes, para la gestión de cada estudiante de forma independiente se realizará mediante un ArrayList con el fin de tener el siguiente esquema

Donde el HashMap tendrá como clave el documento de identidad de la persona y como valor una lista con la información de las personas, a su vez esta lista será de tipo String para permitir almacenar los diferentes tipos de datos y ya luego se realizará la conversión a los datos numéricos al momento de requerirse.

Nota: Este proceso puede hacerse más optimo al trabajar con Objetos, sin embargo para este caso trabajaremos con Listas con el fin de fortalecer la lógica de programación.

Para este proceso se crearán 3 clases, una clase Principal que llamará a la clase procesos y una tercer clase que contendrá el modelo de datos.



Principal.java

Clase principal que hace el llamado a la clase Procesos.

```
public class Principal {

    public static void main(String[] args) {
        new Procesos();
    }
}
```

ModeloDatos.java

Esta clase tendrá el modelo de datos del sistema que corresponde a nuestro HashMap, la idea de esta clase es tener independencia en nuestro modelo de datos para no mezclar la clase procesos con la estructura de datos, dando mayor organización al centralizar aquí lo relacionado con la estructura de datos principal.

```
import java.util.ArrayList;
import java.util.HashMap;
import java.util.Iterator;

import javax.swing.JOptionPane;

public class ModeloDatos {


    HashMap<String, ArrayList<String>> mapEstudiantes;

    public ModeloDatos() {
        mapEstudiantes=new HashMap<String, ArrayList<String>>();

    }

    public void consultarEstudiante(String documento) {

        if (mapEstudiantes.containsKey(documento)) {

            System.out.println(mapEstudiantes.get(documento));

    }else {

        JOptionPane.showMessageDialog(null, "Documento no existe",
                "Advertencia",JOptionPane.WARNING_MESSAGE);
    }

}

public HashMap<String, ArrayList<String>> getMapaEstudiantes(){
    return mapEstudiantes;
}

public void guardarDatos(ArrayList<String> listaEstudiantes) {
    mapEstudiantes.put(listaEstudiantes.get(0), listaEstudiantes);
    JOptionPane.showMessageDialog(null, "Registro existoso!");
}

public int getSizeMap() {
```

```
        return mapEstudiantes.size();
    }

/*
* Otra forma de imprimir un HashMap
*/
    public void imprimirMapaForma2() {

        for (String documento:mapEstudiantes.keySet()) {
            System.out.println(documento + " - "+mapEstudiantes.get(documento));
        }

    }

    public void imprimirMapa() {
        Iterator<String> itera=mapEstudiantes.keySet().iterator();
        String llave;
        while (itera.hasNext()) {
            llave =itera.next();
            System.out.println(llave + " - "+mapEstudiantes.get(llave));
        }
    }

}
```

Procesos.java

Clase donde se realizara la lógica de negocio, desde aquí se hace la instancia de la clase ModeloDatos.java con el fin de acceder a la estructura de datos y tener mucho más organizado nuestro código y la gestión de información.

Adicional se crea el ArrayList que almacenará la información de cada persona, note que la instancia de hace cada vez que se registra un nuevo elemento, esto para garantizar que en el HashMap se guarde una lista por cada estudiante.

Noten que para gestionar la información se hace uso del objeto miData que internamente tiene el HashMap, de esta manera si queremos registrar un nuevo estudiante, tan solo se llama al método de registro de miData.

```
import java.util.ArrayList;
import java.util.HashMap;
import java.util.Iterator;
import javax.swing.JOptionPane;

public class Procesos {

    ModeloDatos miData;
    ArrayList<String> listEstudiantes;
    int cantGanan=0,cantRecuperan=0,cantPierden=0;

    public Procesos() {

        miData=new ModeloDatos();

        iniciar();
    }

    private void iniciar() {

        String resp="";
```

```
    do {

        ingresarDatos();

        resp=JOptionPane.showInputDialog("Ingrese si, para continuar registrando estudiantes");
    }while(resp.equalsIgnoreCase("si"));

    //System.out.println(mapEstudiantes);
    miData.imprimirMapa();
    System.out.println();
    miData.imprimirMapaForma2();
    imprimirResultados();

    String documento=JOptionPane.showInputDialog("Ingrese el documento a buscar");
    miData.consultarEstudiante(documento);
}

private void ingresarDatos() {

    listEstudiantes=new ArrayList<String>();

    String documento=JOptionPane.showInputDialog("Ingrese el documento");
    String nombre=JOptionPane.showInputDialog("Ingrese el nombre del estudiante");
    String materia=JOptionPane.showInputDialog("Ingrese la materia del estudiante");
    double nota1=validarNotas("Ingrese la nota 1");
    double nota2=validarNotas("Ingrese la nota 2");
    double nota3=validarNotas("Ingrese la nota 3");

    listEstudiantes.add(documento);
    listEstudiantes.add(nombre);
    listEstudiantes.add(materia);
    listEstudiantes.add(nota1+"");
    listEstudiantes.add(nota2+"");
    listEstudiantes.add(nota3+"");

    miData.guardarDatos(listEstudiantes);

    System.out.println("Nombre: "+nombre+", Documento: "+documento+", Materia: "+materia);

    calcularPromedio(nota1,nota2,nota3);

}

public double validarNotas(String msj) {
    double nota=0;

    do {
        nota=Double.parseDouble(JOptionPane.showInputDialog(msj));
        if (nota<0 || nota>5) {
            JOptionPane.showMessageDialog(null,"Nota incorrecta, "
                    + "no esta en el rango, repitala","ERROR",JOptionPane.ERROR_MESSAGE);
        }

    } while (nota<0 || nota>5);


    return nota;
```

```
}

public void imprimirResultados() {

    double sumaTotal=sumaTotalNotas();
    int cantNotas=miData.getSizeMap()*3;

    System.out.println("Cantidad total estudiantes: "+miData.getSizeMap());
    System.out.println("Cantidad total notas ingresadas: "+cantNotas);
    System.out.println("Suma total de notas: "+sumaTotal);
    System.out.println("promedio total de notas: "+(sumaTotal/cantNotas));
    System.out.println("Promedio total de notas finales: "+calcularPromedioFinal());
    System.out.println("Cantidad que ganan: "+cantGanan);
    System.out.println("Cantidad que recuperan: "+cantRecuperan);
    System.out.println("Cantidad que pierden: "+cantPierden);
    System.out.println();
}


private double sumaTotalNotas() {

Iterator<String> itera=miData.getMapaEstudiantes().keySet().iterator();

    double suma=0;

    while (itera.hasNext()) {
        String doc = itera.next();

        ArrayList<String> listTemp=miData.getMapaEstudiantes().get(doc);

        for (int i = 3; i <listTemp.size(); i++) {
            suma+=Double.parseDouble(listTemp.get(i));
        }
    }

    return suma;
}

private double calcularPromedioFinal() {

    Iterator<String> itera=miData.getMapaEstudiantes().keySet().iterator();

    double n1,n2,n3;
    double suma=0,prom;

    while (itera.hasNext()) {
        String doc = itera.next();

        ArrayList<String> listTemp=miData.getMapaEstudiantes().get(doc);

        n1=Double.parseDouble(listTemp.get(3));
        n2=Double.parseDouble(listTemp.get(4));
        n3=Double.parseDouble(listTemp.get(5));

        prom=(n1+n2+n3)/3;
        calcularEstados(prom);
        suma+=prom;
```

```
    }

    return suma/miData.getSizeMap();
}

public void calcularEstados(double promedio) {

    if (promedio>=3.5) {
        cantGanan++;
    }else {
        if (promedio>=2) {
            cantRecuperan++;
        }else {
            cantPierden++;
        }
    }
}

    private void calcularPromedio(double nota1, double nota2, double nota3) {

        double promedio=(nota1+nota2+nota3)/3;

        System.out.println("("+nota1+"+"+nota2+"+"+nota3+")/3="+promedio);

        if (promedio>=3.5) {
            System.out.println("Gana la materia");
        }else {
            if (promedio>=2) {
                System.out.println("Pierde, pero puede recuperar");
            }else {
                System.out.println("Pierde la materia");
            }
        }
        System.out.println();
    }

}
```

Actividad.

1. Transcriba el código y analice su comportamiento.
2. Documente el código explicando el funcionamiento.
3. Cree un menú que permita gestionar el sistema, de la siguiente forma:
    1. Registrar otro estudiante.
    2. Consultar estudiante por documento
    3. Consultar Lista de estudiantes
    4. Cantidad total de estudiantes validados
    5. Cantidad total de notas ingresadas
    6. Sumatoria total de las notas ingresadas
    7. Promedio de las notas finales obtenidas.
    8. Cantidad de estudiantes qué ganan la materia (ganan cuando su promedio es mayor o igual a 3.5)
    9. Cantidad de estudiantes qué pierden la materia (pierden cuando su promedio es menor a 3.5)

10. Cantidad de estudiantes qué pueden recuperar (cuando su nota final es superior a 2)
11. Cantidad de estudiantes qué pierden sin recuperación.