**ENUNCIADO DEL ENSAYO**

**El aula de clases**

Queremos representar con programación orientada a objetos, un aula con estudiantes y un profesor, dados los siguientes requisitos:

* Tanto de los estudiantes como de los profesores necesitamos saber su nombre, edad y sexo. De los estudiantes, queremos saber también su calificación actual *(entre 0 y 40)* y del profesor que materia da.
* Las materias disponibles son matemáticas, filosofía y física.
* Los estudiantes tendrán un 50% de *hacer novillos* *(hacer novillos es lo mismo que decir, faltar a clase)*.
* El profesor tiene un 20% de no encontrarse disponible *(no ir a clases por reuniones, salud, etc.).*
* Las dos operaciones anteriores *(hacer novillos y/o inasistencia del docente)* deben llamarse igual en Estudiante y Profesor *(polimórficamente)*.
* El aula debe tener un identificador numérico, el número máximo de estudiantes y para que esta destinada *(matemáticas, filosofía o física)*. Si Ud. considera más atributos, tiene la libertar de hacerlo.
* En un aula, para que se pueda dar clase se necesita que el profesor esté disponible, es decir, un profesor de la materia correspondiente en el aula correspondiente *(ejemplo: un profesor de filosofía no puede dar en un aula de matemáticas), además*, que haya al menos un 50% de alumnos matriculados del máximo definido para dicha aula.

El objetivo es crear un aula de alumnos y un profesor y determinar si puede darse clase, teniendo en cuenta las condiciones antes dichas. Si se puede dar clase mostrar cuantos alumnos y alumnas *(por separado)* están aprobado *(un estudiante aprueba si tiene un 50% de asistencias, y como mínimo 28/40 pt.)*

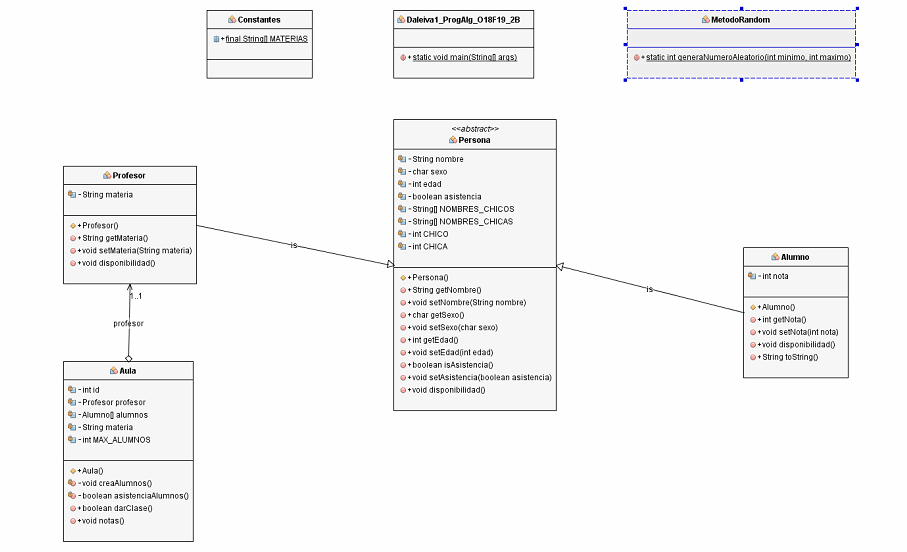
**Se pide:**

* Modelar la jerarquía de clases *con herencia en UML* para el escenario antes descrito; así como la clase (***class***) de ***Prueba*** que permita realizar la creación a Aulas Clase.
* Codificar la jerarquía de clases *(UML antes diseñado en el informe* ***.docx****)* que crea conveniente aplicando/aprovechando al máximo las técnicas de Herencia*.*
* Los datos para la interacción *usuario/programa* deben autogenerarse con la clase **Random**; cuando su docente vaya a revisar su aplicación, no se debe solicitar el ingreso de dato alguno, todos los datos que requiera su app. deben ser autogenerados aleatoriamente, su docente validará los datos de salida o resultados, en el archivo de salida ***suUsername\_DatosSalida.csv***, los cuales serán una copia de lo visualizado en consola, *(listado de Aulas con los Profesores, y estudiantes matriculados)*. Este archivo ***.csv***, debe escribirse usando rutas relativas, con el fin de evitar conflictos de escritura, permisos, etc. que se dan con las rutas absolutas.
* Debe *instanciar/probar* esta jerarquía de clases en una clase **CLASE\_DE\_PRUEBA** *(como lo hacen los ejemplos desarrollados en su libro base)* en la cual se instancie las clases, se envíe los valores *(auto-generados o inicializados)*, realicen los cálculos y se generen los resultados *(por consola y hacia el archivo de salida con extensión* ***.csv*** *– archivos de texto editables también con Excel)*.
* Los datos de entrada y resultados deben ser comprensibles, tanto hacia consola como hacia el archivo de salida ***.csv***; por lo tanto, queda a su elección presentar dichos datos de entrada/salida de una manera creativa y usable.

***Importante:*** Si en el transcurso del desarrollo de este trabajo a distancia Ud. tuviera inconvenientes, con todo gusto le atenderemos en los medios compartidos *(por ejemplo, foros)* que su Tutor disponga en la plataforma virtual de aprendizaje.

## **Análisis y diseño de la solución *(Diagrama de clases)***

*A continuación, un ejemplo de cómo se debe presentar el diagrama de clase UML, el cual Ud. lo puede generar con cualquier herramienta (NetBeans, Eclipse, IntelliJ IDEA, etc.), caso contrario, Visio, o el mismo Word son aplicables. Dicho diagrama es obligatorio, por cuanto su* ***.java*** *debe aplicar conceptos de* ***Herencia****. Borrar este comentario al redactar su informe.*



**Documentación del diagrama de clases UML:**

|  |  |  |
| --- | --- | --- |
| Nombre de la Clase | Descripción | |
| *Clase Constante* | La clase ***<*** *Clase Constante* ***>*** tiene definido el vector Materia, con los datos de las materias. | |
| **Métodos** | **Descripción** |
| ***Constantes*** | Definición de vector |
|  |  |
| *Clase MetodoRandom* | La clase ***<*** *Clase MetodoRandom* ***>*** realiza los números aleatorios enviando dos parámetros. | |
| **Métodos** | **Descripción** |
| ***generaNumeroAleatorio*** | Genera números aleatorios |
|  |  |
| *Clase*  *Dennis1\_ProgAlf\_I18F19\_2b* | La clase ***<*** *Dennis1\_ProgAlf\_I18F19\_2b* ***>*** permite la ejecución al realizar la llamada a la clase aula | |
| **Métodos** | **Descripción** |
| ***MetodoPRINCIPAL*** | Este método es el principal donde esta construido el case para seleccionar los casos |
|  |  |
| *Clase* Persona | La clase abstracto ***<*** *Clase* Persona ***>*** permite unificar datos u operaciones de subclases | |
| **Métodos** | **Descripción** |
| ***getNombre*** | Devuelve el nombre |
| ***setNombre*** | Modifica el nombre |
| ***getSexo*** | Devuelve el sexo de la persona |
| ***setSexo*** | Modifica el sexo de la persona |
| ***getEdad*** | Devuelve la edad de la persona |
| ***setEdad*** | Modifica la edad de la edad |
|  | ***isAsistencia*** | Indica la asistencia de la persona |
|  | ***setAsistencia*** | Modifica la asistencia de la persona |
| *Clase Profesor* | La clase ***<*** *Clase Profesor* ***>*** hereda de la súper clase ***< Persona >*** para permitir procesar los datos. | |
| **Métodos** | **Descripción** |
| ***getMateria*** | Devuelve la materia del profesor |
| ***setMateria*** | Modifica la materia del profesor |
|  | ***disponibilidad*** | Calcula la disponibilidad del profesor (20%) |
| *Clase Alumno* | La clase *<* *Clase Alumno >* hereda de la súper clase ***< Persona >*** para permitir procesar los datos. | |
| **Métodos** | **Descripción** |
| ***getNota*** | Devuelve la nota |
| ***setNota*** | Modifica la nota del alumno |
|  | ***disponibilidad*** | Indica si el alumno está disponible (50%) |
|  | ***toString*** | Muestra la información del alumno |
| *Clase Aula* | La clase ***<*** *Clase Aula****>*** crea la información de alumnos, notas, genera la asistencia. | |
| **Métodos** | **Descripción** |
| **creaAlumnos** | Crea los alumnos para el aula |
| ***asistenciaAlumnos*** | Indica si la asistencia de los alumnos es mayor del 50% |
| ***darClase*** | Indicamos si se puede dar la clase |
| ***notas*** | Indicamos las notas de los alumnos aprobados, chicos y chicas |

**Desarrollo de la solución *daleiva1\_ProgAlg\_AA19\_2B.java***

1. **package** daleiva1\_progalg\_o18f19\_2b;
3. **import** java.io.IOException;
5. ***/\*\****
6. ***\****
7. ***\* @author Dennis Leiva***
8. ***\*/***
9. **public** **class** Daleiva1\_ProgAlg\_O18F19\_2B {
11. ***/\*\****
12. ***\* @param args the command line arguments***
13. ***\*/***
14. **public** **static** **void** main([String](http://www.google.com/search?hl=en&q=allinurl%3Astring+java.sun.com&btnI=I%27m%20Feeling%20Lucky)[] args) **throws** [IOException](http://www.google.com/search?hl=en&q=allinurl%3Aioexception+java.sun.com&btnI=I%27m%20Feeling%20Lucky)  {
15. *//Creamos el objeto*
16. **Aula aula=new Aula();**
18. *//Indicamos si se puede dar la clase*
19. **if**(aula.darClase()){
20. aula.notas();
22. }
24. }
26. }
28. ***/\****
29. *\* To change this license header, choose License Headers in Project Properties.*
30. *\* To change this template file, choose Tools | Templates*
31. *\* and open the template in the editor.*
32. *\*/*
33. **package daleiva1\_progalg\_o18f19\_2b;**
35. ***/\*\****
36. ***\****
37. ***\* @author Dennis Leiva***
38. ***\*/***
39. **public** **class** Constantes {
41. **public** **static** **final** [String](http://www.google.com/search?hl=en&q=allinurl%3Astring+java.sun.com&btnI=I%27m%20Feeling%20Lucky)[] MATERIAS={"Matematicas", "Filosofia", "Fisica"};


45. }

48. ***/\****
49. *\* To change this license header, choose License Headers in Project Properties.*
50. *\* To change this template file, choose Tools | Templates*
51. *\* and open the template in the editor.*
52. *\*/*
53. **package daleiva1\_progalg\_o18f19\_2b;**
55. ***/\*\****
56. ***\****
57. ***\* @author Dennis Leiva***
58. ***\*/***
59. **public** **class** MetodoRandom {
61. ***/\*\****
62. ***\* Genera un numero aleatorio entre dos numeros.***
63. ***\* Entre el minimo y el maximo***
64. ***\* @param minimo***
65. ***\* @param maximo***
66. ***\* @return numero entre minimo y maximo***
67. ***\*/***
68. **public static int generaNumeroAleatorio(int minimo, int maximo){**
70. **int** num=(**int**)[Math](http://www.google.com/search?hl=en&q=allinurl%3Amath+java.sun.com&btnI=I%27m%20Feeling%20Lucky).floor([Math](http://www.google.com/search?hl=en&q=allinurl%3Amath+java.sun.com&btnI=I%27m%20Feeling%20Lucky).random()\*(minimo-(maximo+1))+(maximo+1));
71. **return** num;
72. }

75. }

78. ***/\****
79. *\* To change this license header, choose License Headers in Project Properties.*
80. *\* To change this template file, choose Tools | Templates*
81. *\* and open the template in the editor.*
82. *\*/*
83. **package daleiva1\_progalg\_o18f19\_2b;**
85. ***/\*\****
86. ***\****
87. ***\* @author Dennis Leiva***
88. ***\*/***
89. **public** **abstract** **class** Persona {
91. */\*Atributos\*/*
92. **private** [String](http://www.google.com/search?hl=en&q=allinurl%3Astring+java.sun.com&btnI=I%27m%20Feeling%20Lucky) nombre;
93. **private char sexo;**
94. **private** **int** edad;
95. **private** **boolean** asistencia;
97. */\*Contantes\*/*
98. **private final**[**String**](http://www.google.com/search?hl=en&q=allinurl%3Astring+java.sun.com&btnI=I%27m%20Feeling%20Lucky)**[] NOMBRES\_CHICOS={"Gorki", "Rubén", "Eloy", "Diego", "Javier"};**
99. **private** **final** [String](http://www.google.com/search?hl=en&q=allinurl%3Astring+java.sun.com&btnI=I%27m%20Feeling%20Lucky)[] NOMBRES\_CHICAS={"Yady", "Danna", "Analia", "Francisca", "Johanna"};
100. **private** **final** **int** CHICO=0;
101. **private** **final** **int** CHICA=1;
103. ***/\*Constructores\*/***
104. **public** Persona(){
106. *//entre 0 y 1*
107. **int** determinar\_sexo=MetodoRandom.generaNumeroAleatorio(0,1);
109. *//Si es 0 es un chico*
110. **if**(determinar\_sexo==CHICO){
111. nombre=NOMBRES\_CHICOS[MetodoRandom.generaNumeroAleatorio(0,4)];
112. sexo='H';
113. **}else{**
114. nombre=NOMBRES\_CHICAS[MetodoRandom.generaNumeroAleatorio(0,4)];
115. sexo='M';
116. }
118. ***//Indicamos la disponibilidad***
119. disponibilidad();
121. }
123. ***/\*Metodos\*/***
125. ***/\*\****
126. ***\* Devuelve el nombre***
127. ***\* @return***
128. ***\*/***
129. **public** [String](http://www.google.com/search?hl=en&q=allinurl%3Astring+java.sun.com&btnI=I%27m%20Feeling%20Lucky) getNombre() {
130. **return** nombre;
131. }
133. ***/\*\****
134. ***\* Modifica el nombre***
135. ***\* @param nombre***
136. ***\*/***
137. **public** **void** setNombre([String](http://www.google.com/search?hl=en&q=allinurl%3Astring+java.sun.com&btnI=I%27m%20Feeling%20Lucky) nombre) {
138. **this.nombre = nombre;**
139. }
141. ***/\*\****
142. ***\* Devuelve el sexo de la persona***
143. ***\* @return***
144. ***\*/***
145. **public** **char** getSexo() {
146. **return** sexo;
147. }
149. ***/\*\****
150. ***\* Modifica el sexo de la persona***
151. ***\* @param sexo***
152. ***\*/***
153. **public void setSexo(char sexo) {**
154. **this**.sexo = sexo;
155. }
157. ***/\*\****
158. ***\* Devuelve la edad de la persona***
159. ***\* @return***
160. ***\*/***
161. **public** **int** getEdad() {
162. **return** edad;
163. **}**
165. ***/\*\****
166. ***\* Modifica la edad de la edad***
167. ***\* @param edad***
168. ***\*/***
169. **public** **void** setEdad(**int** edad) {
170. **this**.edad = edad;
171. }
173. ***/\*\****
174. ***\* Indica la asistencia de la persona***
175. ***\* @return***
176. ***\*/***
177. **public** **boolean** isAsistencia() {
178. **return asistencia;**
179. }
181. ***/\*\****
182. ***\* Modifica la asistencia de la persona***
183. ***\* @param asistencia***
184. ***\*/***
185. **public** **void** setAsistencia(**boolean** asistencia) {
186. **this**.asistencia = asistencia;
187. }
189. *//abtracto, las clases hijas deben implementarlo*
190. **public** **abstract** **void** disponibilidad();
192. }
194. \*
195. \* To change **this** license header, choose License Headers in Project [Properties](http://www.google.com/search?hl=en&q=allinurl%3Aproperties+java.sun.com&btnI=I%27m%20Feeling%20Lucky).
196. \* To change **this** template file, choose Tools | Templates
197. \* and open the template in the editor.
198. **\*/**
199. **package** daleiva1\_progalg\_o18f19\_2b;
201. ***/\*\****
202. ***\****
203. ***\* @author Dennis Leiva***
204. ***\*/***
205. **public** **class** Profesor **extends** Persona {
207. */\*Atributos\*/*
208. **private [String](http://www.google.com/search?hl=en&q=allinurl%3Astring+java.sun.com&btnI=I%27m%20Feeling%20Lucky) materia;**
210. */\*Constructores\*/*
211. **public** Profesor(){
212. **super**(); *//Llama al constructor padre*
214. **super**.setEdad(MetodoRandom.generaNumeroAleatorio(25,50)); *//llama al metodo padre*
216. materia=Constantes.MATERIAS[MetodoRandom.generaNumeroAleatorio(0,2)];
217. }
219. */\*Metodos\*/*
221. ***/\*\****
222. ***\* Devuelve la materia del profesor***
223. ***\* @return***
224. ***\*/***
225. **public** [String](http://www.google.com/search?hl=en&q=allinurl%3Astring+java.sun.com&btnI=I%27m%20Feeling%20Lucky) getMateria() {
226. **return** materia;
227. }
229. ***/\*\****
230. ***\* Modifica la materia del profesor***
231. ***\* @param materia***
232. ***\*/***
233. **public void setMateria([String](http://www.google.com/search?hl=en&q=allinurl%3Astring+java.sun.com&btnI=I%27m%20Feeling%20Lucky) materia) {**
234. **this**.materia = materia;
235. }
237. ***/\*\****
238. ***\* Calcula la disponibilidad del profesor(20%)***
239. ***\*/***
240. @Override
241. **public** **void** disponibilidad() {
243. **int prob=MetodoRandom.generaNumeroAleatorio(0, 100);**
245. **if**(prob<20){
246. **super**.setAsistencia(**false**);
247. }**else**{
248. **super.setAsistencia(true);**
249. }
251. }
253. **}**
255. */\**
256. *\* To change this license header, choose License Headers in Project Properties.*
257. *\* To change this template file, choose Tools | Templates*
258. ***\* and open the template in the editor.***
259. *\*/*
260. **package** daleiva1\_progalg\_o18f19\_2b;
262. ***/\*\****
263. ***\****
264. ***\* @author Dennis Leiva***
265. ***\*/***
266. **public** **class** Alumno **extends** Persona {
268. ***/\*Atributos\*/***
269. **private** **int** nota;
271. */\*Constructor\*/*
272. **public** Alumno(){
273. **super();**
275. nota=MetodoRandom.generaNumeroAleatorio(0,40);
277. **super**.setEdad(MetodoRandom.generaNumeroAleatorio(12,15));
279. }
281. */\*Metodos\*/*
283. ***/\*\****
284. ***\* Devuelve la nota***
285. ***\* @return nota del alumno***
286. ***\*/***
288. **public int getNota() {**
289. **return** nota;
290. }
292. ***/\*\****
293. ***\* Modifica la nota del alumno***
294. ***\* @param nota***
295. ***\*/***
296. **public** **void** setNota(**int** nota) {
297. **this**.nota = nota;
298. **}**
300. ***/\*\****
301. ***\* Indica si el alumno esta disponible (50%)***
302. ***\*/***
303. **@Override**
304. **public** **void** disponibilidad() {
306. **int** prob=MetodoRandom.generaNumeroAleatorio(0, 100);
308. **if(prob<50){**
309. **super**.setAsistencia(**false**);
310. }**else**{
311. **super**.setAsistencia(**true**);
312. }
314. }
316. ***/\*\****
317. ***\* Muestra la informacion del alumno***
318. ***\* @return informacion***
319. ***\*/***
320. **public** [String](http://www.google.com/search?hl=en&q=allinurl%3Astring+java.sun.com&btnI=I%27m%20Feeling%20Lucky) toString(){

323. **return "Nombre: "+super.getNombre()+" ,sexo: "+super.getSexo()+" , nota: "+nota;**



328. **}**

331. }
333. ***/\****
334. *\* To change this license header, choose License Headers in Project Properties.*
335. *\* To change this template file, choose Tools | Templates*
336. *\* and open the template in the editor.*
337. *\*/*
338. **package daleiva1\_progalg\_o18f19\_2b;**
340. **import** com.csvreader.CsvWriter;
341. **import** java.io.IOException;
342. ***/\*\****
343. ***\****
344. ***\* @author Familia***
345. ***\*/***
346. **public** **class** Aula {

349. */\*Atributos\*/*
350. **private** **int** id;
351. **private** Profesor profesor;
352. **private** Alumno[] alumnos;
353. **private [String](http://www.google.com/search?hl=en&q=allinurl%3Astring+java.sun.com&btnI=I%27m%20Feeling%20Lucky) materia;**
355. */\*Constantes\*/*
356. **private** **final** **int** MAX\_ALUMNOS=20;
358. ***/\*Constructores\*/***
359. **public** Aula(){
361. id=1;
363. **profesor=new Profesor();**
364. alumnos= **new** Alumno[MAX\_ALUMNOS];
365. creaAlumnos();
366. materia=Constantes.MATERIAS[MetodoRandom.generaNumeroAleatorio(0,2)];
368. **}**
370. */\*Metodos\*/*
372. ***/\*\****
373. ***\* Crea los alumnos para el aula***
374. ***\*/***
375. **private** **void** creaAlumnos(){
377. **for**(**int** i=0;i<alumnos.length;i++){
378. **alumnos[i]=new Alumno();**
379. }
381. }

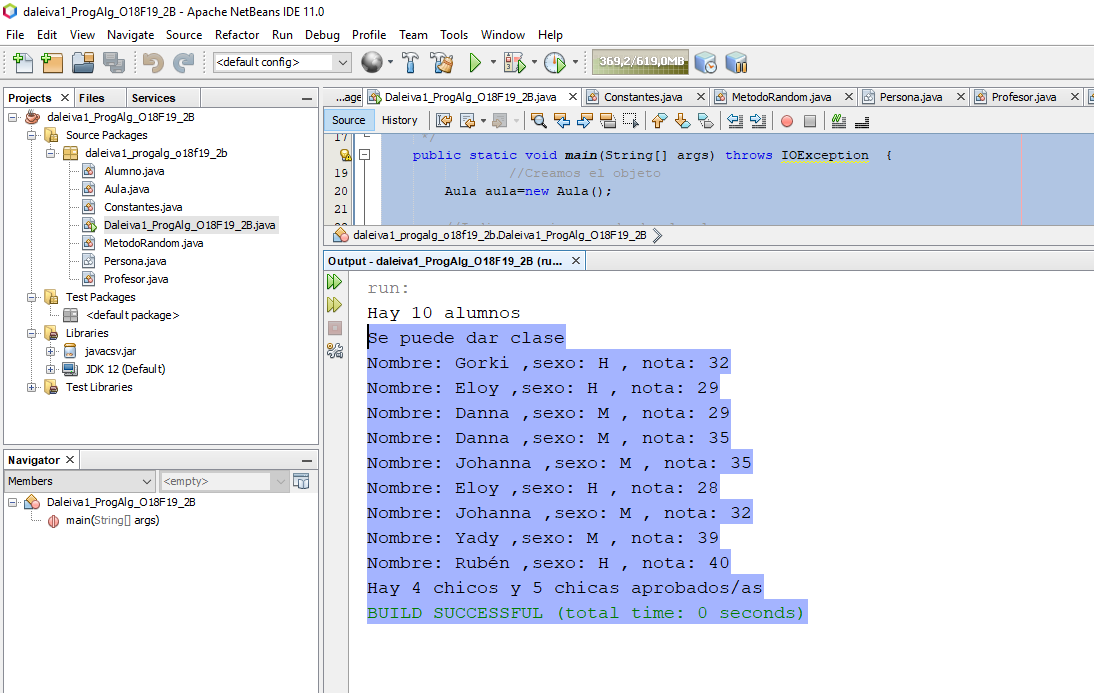
384. ***/\*\****
385. ***\* Indica si la asistencia de los alumnos es mayor del 50%***
386. ***\* @return***
387. ***\*/***
388. **private boolean asistenciaAlumnos(){**
390. **int** cuentaAsistencias=0;
392. *//contamos las asistencias*
393. **for(int i=0;i<alumnos.length;i++){**
395. **if**(alumnos[i].isAsistencia()){
396. cuentaAsistencias++;
397. }
399. }
401. *//Muestro la asistencia total*
402. [System](http://www.google.com/search?hl=en&q=allinurl%3Asystem+java.sun.com&btnI=I%27m%20Feeling%20Lucky).out.println("Hay "+cuentaAsistencias+" alumnos");
404. **return** cuentaAsistencias>=((**int**)(alumnos.length/2));
406. }
408. ***/\*\****
409. ***\* Indicamos si se puede dar la clase***
410. ***\* @return***
411. ***\*/***
412. **public** **boolean** darClase() **throws** [IOException](http://www.google.com/search?hl=en&q=allinurl%3Aioexception+java.sun.com&btnI=I%27m%20Feeling%20Lucky){
413. **int cuentaAsistencias=0;**
415. *//contamos las asistencias*
416. **for**(**int** i=0;i<alumnos.length;i++){
418. **if(alumnos[i].isAsistencia()){**
419. cuentaAsistencias++;
420. }
422. }
423. ***//Indicamos las condiciones para que se pueda dar la clase***
424. CsvWriter csvWriter = **new** CsvWriter("fichero.Csv");
425. **if**(!profesor.isAsistencia()){
426. [System](http://www.google.com/search?hl=en&q=allinurl%3Asystem+java.sun.com&btnI=I%27m%20Feeling%20Lucky).out.println("El profesor no esta, no se puede dar clase");
427. ***/\*\****
428. ***\* se exporta a csv***
429. ***\*/***
430. csvWriter.write("El profesor no esta, no se puede dar clase");
431. csvWriter.close();
432. **return** **false**;
433. **}else if(!profesor.getMateria().equals(materia)){**
434. [System](http://www.google.com/search?hl=en&q=allinurl%3Asystem+java.sun.com&btnI=I%27m%20Feeling%20Lucky).out.println("La materia del profesor y del aula no es la misma, no se puede dar clase");
435. ***/\*\****
436. ***\* se exporta a csv***
437. ***\*/***
439. csvWriter.write("La materia del profesor y del aula no es la misma, no se puede dar clase");
440. csvWriter.close();
441. **return** **false**;
442. }**else** **if** (!asistenciaAlumnos()){
443. **[System](http://www.google.com/search?hl=en&q=allinurl%3Asystem+java.sun.com&btnI=I%27m%20Feeling%20Lucky).out.println("La asistencia no es suficiente, no se puede dar clase");**
444. ***/\*\****
445. ***\* se exporta a csv***
446. ***\*/***
447. csvWriter.write("Hay "+cuentaAsistencias+" alumnos");
448. **csvWriter.endRecord();**
449. csvWriter.write("La asistencia no es suficiente, no se puede dar clase");
450. csvWriter.close();
451. **return** **false**;
452. }
454. [System](http://www.google.com/search?hl=en&q=allinurl%3Asystem+java.sun.com&btnI=I%27m%20Feeling%20Lucky).out.println("Se puede dar clase");



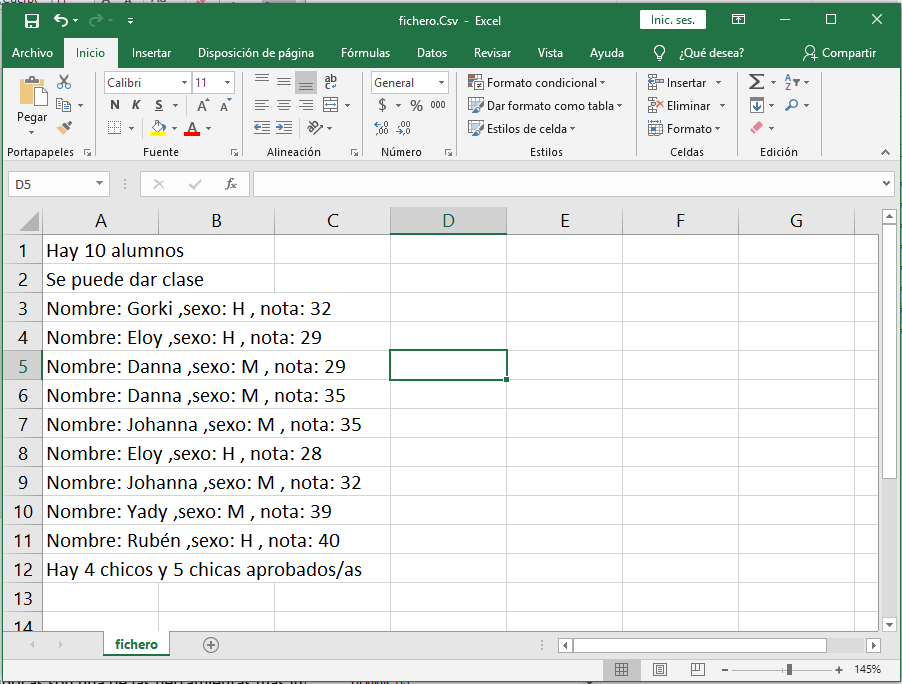
459. **return** **true**;
461. }
463. ***/\*\****
464. ***\* Indicamos las notas de los alumnos aprobados, chicos y chicas***
465. ***\*/***
466. **public** **void** notas() **throws** [IOException](http://www.google.com/search?hl=en&q=allinurl%3Aioexception+java.sun.com&btnI=I%27m%20Feeling%20Lucky){
468. **int chicosApro=0;**
469. **int** chicasApro=0;
471. **int** cuentaAsistencias=0;
473. ***//contamos las asistencias***
474. **for**(**int** i=0;i<alumnos.length;i++){
476. **if**(alumnos[i].isAsistencia()){
477. cuentaAsistencias++;
478. **}**
480. }



485. ***/\*\****
486. ***\* se exporta a csv***
487. ***\*/***
488. **CsvWriter csvWriter = new CsvWriter("fichero.Csv");**
489. ***/\*\****
490. ***\* se exporta a csv***
491. ***\*/***
492. csvWriter.write("Hay "+cuentaAsistencias+" alumnos");
493. **csvWriter.endRecord();**
494. csvWriter.write("Se puede dar clase");
495. csvWriter.endRecord();
497. **for**(**int** i=0;i<alumnos.length;i++){
499. *//Comprobamos si el alumno esta aprobado*
500. **if**(alumnos[i].getNota()>=28){
501. *//Segun el sexo, aumentara uno o otro*
502. **if**(alumnos[i].getSexo()=='H'){
503. **chicosApro++;**
504. }**else**{
505. chicasApro++;
506. }
508. **[System](http://www.google.com/search?hl=en&q=allinurl%3Asystem+java.sun.com&btnI=I%27m%20Feeling%20Lucky).out.println(alumnos[i].toString());**
510. ***/\*\****
511. ***\* se exporta a csv***
512. ***\*/***
513. **csvWriter.write(alumnos[i].toString());**
515. csvWriter.endRecord();
517. }
518. }
520. [System](http://www.google.com/search?hl=en&q=allinurl%3Asystem+java.sun.com&btnI=I%27m%20Feeling%20Lucky).out.println("Hay "+chicosApro+" chicos y "+chicasApro+" chicas aprobados/as");
522. ***/\*\****
523. ***\* se exporta a csv***
524. ***\*/***
525. csvWriter.write("Hay "+chicosApro+" chicos y "+chicasApro+" chicas aprobados/as");
526. **csvWriter.close();**
528. }
530. }

**Run aplicativo** 

**Archivo csv**



**RÚBRICA DE EVALUACIÓN**

Las rúbricas son una de las herramientas más importantes para evaluar de una manera objetiva, precisa y realista, es oportuno validar los ensayos con criterios bien definidos calificando el esfuerzo y resultados de todo un bimestre, evidenciando así el nivel de competencias adquiridas en la asignatura de **Programación de Algoritmos**.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| ***ASPECTOS / ESCALA*** | **Excelente** | **satisfactorio** | **en progreso** | **iniciado** | **Ausente** | ***Puntaje*** |
|  | **4 pts.** | **3 pts.** | **2 pts.** | **1 pt.** | **0 pts.** |  |
| **Legibilidad y Documentación** | La estructura del código es bastante clara y limpia, con suficiente documentación / comentarios | La estructura del código es clara y limpia, con documentación / comentarios | La estructura del código no es muy clara ni limpia, la documentación es mínima | La estructura del código es confusa, casi no documenta ni comenta. | Código sin estructura, totalmente confuso, sin documentación ni comentarios | |  | | --- | | **\_\_ / 4pts.** | |
| **Datos de entrada** | Definidos, ingresados y/o auto generados de forma totalmente clara y organizada | Definidos, ingresados y/o auto generados de forma clara y organizada | Definidos, ingresados y/o auto generados de manera parcialmente clara y con escaso orden | Definidos, ingresados y/o auto generados de forma totalmente confusa, ausente y/o sin orden | No define, ingresa y/o auto genera datos de entrada alguno | |  | | --- | | **\_\_ / 4pts.** | |
| **Resultados** | Procesados y visualizados de forma totalmente clara y organizada | Procesados y visualizados de forma clara y organizada | Procesados y visualizados de manera parcialmente clara y con escaso orden | Casi no procesa ni visualiza resultado alguno | No procesa ni visualiza resultado alguno | |  | | --- | | **\_\_ / 4pts.** | |
| **Resolución del problema** | Resuelve todos los requerimientos con una lógica totalmente comprensible | Resuelve los requerimientos con una lógica comprensible | Resuelve algunos requerimientos con una lógica parcialmente comprensible | No resuelve todos los requerimientos, su lógica es confusa | No resuelve ningún requerimiento y/o lógica totalmente confusa | |  | | --- | | **\_\_ / 4pts.** | |
| **Funcionamiento** | La solución compila y ejecuta de forma sencilla, según las especificaciones | La solución compila y ejecuta según las especificaciones | La solución no compila y/o ejecuta de forma sencilla, según las especificaciones | La solución no compila y/o ejecuta | No adjunta la solución para su compilación y/o ejecución | |  | | --- | | **\_\_ / 4pts.** | |
| **Informe** | Es totalmente claro y organizado, guardando formatos e información solicitada | Es claro y/u organizado con formatos e información solicitada | Es parcialmente claro y organizado, casi no guarda formatos y/o información solicitada | Es totalmente confuso, sin formatos e información solicitada | No adjunta informe solicitado | |  | | --- | | **\_\_ / 4pts.** | |
| **Puntaje total de la rúbrica** según la ***escala*** seleccionada para cada ***aspecto***: | | | | | | **\_\_\_ / 24pts.** |