GESTIÓN DE ALUMNOS, CALIFICACIONES Y ASISTENCIA

Realizar una aplicación que permita realizar la gestión de los alumnos de un centro educativo, administrar las faltas de asistencia y poner calificaciones trimestrales. A continuación se indican los mínimos que deberían tener cada clase implicada. Esto se puede ampliar y crear las clases que sean necesarias para un mejor funcionamiento.

A. PARTE 1

Crea las siguientes clases:

1) Clase **Fecha**

- *Atributos*:
  int dia
  int mes
  int año

- *Métodos*:
  Constructor(int dia, int mes, int año), que inicializa los atributos. Se debe comprobar que los valores día, mes y año son correctos. Tendremos en cuenta que los meses tienen 30, 31 y 28 días (no contemplamos años bisiestos) según el mes. El año tiene que ser posterior a 2015.
  Getters.
  imprimeFecha(), que muestra en pantalla la fecha con el siguiente formato dia/mes/año.
  equals(Object object), sobrecarga el método equals de la clase Object. (Os lo doy hecho).

2) Clase **Horario**

- *Atributos:*
  char [6] sesiones, que representa las 6 horas de clase

- *Métodos*:
  Constructor(), que inicializa todos a ‘ ’.
  GetSesiones(), que devuelve al array de caracteres sesiones.
  faltaDiaEntero(), que pone todas las horas a ‘F’.
  faltaHora (int sesion), que pone la sesión que pasa a ‘F’. Sesión tiene que ser un valor entre 1 y 6.
  imprimeHorario(), que muestra en pantalla el array de sesiones con el siguiente formato F/F/F/ / /F.

3) Clase **DiaClase**

- *Atributos*:
  Fecha dia
  Horario sesiones

- *Métodos*:
  Constructor(Fecha dia), que inicializa el día y crea la instancia de horario.
  Getters.
  SetDia().
  equals(Object object), sobrecarga el método equals de la clase Object. (Os lo doy hecho).

4) Clase **Calificacion**

- *Atributos*:
  String asignatura
  String nota (puede incluir el valor NE, no evaluado).

- *Métodos*:
  Constructor(String asignatura), que inicializa el valor de asignatura.
  Getters y setters.
  equals(Object object), sobrecarga el método equals de la clase Object. (Os lo doy hecho).

5) Clase **Alumno**

- *Atributos*:
  String dni
  String nombre
  String apellidos
  String telefono
  String email
  ArrayList<DiaClase> faltas
  ArrayList<Calificacion> notas

- *Métodos*:
  Constructor(String dni), que asigna dni y crea los ArrayLists.
  Constructor(String dni, String nombre, String apellidos), que asigna dni, nombre y apellidos del alumno y crea los ArrayLists.
  Getters.
  SetDni(), setNombre(), setApellidos(), setTelefono(), setEmail().
  equals(Object object), sobrecarga el método equals de la clase Object. (Os lo doy hecho).

B. PARTE 2

Crea una aplicación funcional que, de forma general, permita gestionar los alumnos del centro (dar de alta, de baja, listar, matricular…), poner y listar calificaciones trimestrales y que permita poner faltas de asistencia tanto en días completos, como en horas sueltas. Para ello crearemos un ***ArrayList de Alumno*** y mostraremos el siguiente menú que permitirá:

1. **Dar de alta alumnos**
2. **Dar de baja alumnos**
3. **Listar los alumnos**
4. **Modificar alumnos**
5. **Matricular alumnos**
6. **Dar de baja de una asignatura**
7. **Introducir calificación trimestral**
8. **Listar calificaciones de alumnos**
9. **Poner una falta (día completo)**
10. **Poner una falta (en una sesión)**
11. **Pasar lista**
12. **Listar faltas**
13. **Salir**

Crea los métodos estáticos que necesites (al menos uno por cada opción del menú).
Cuando sea necesario lanza excepciones y captúralas.

C. PARTE OPCIONAL

Incluye en cada opción del menú (todas menos las opciones 3, 11 y 13) la posibilidad de volver a repetirla.
Por ejemplo: se da de alta un alumno y en lugar de volver a mostrar el menú directamente, se pregunta si se quiere repetir la operación.

D. RECOMENDACIONES

1. Dar de alta alumnos

Se debe comprobar si ya existe el Alumno que se está intentando dar de alta.

2. Dar de baja alumnos

Se debe comprobar que existe el Alumno que se está intentando dar de baja.

3. Listar los alumnos

Se debe comprobar si hay alumnos.

4. Modificar alumnos

Se debe comprobar que existe el Alumno antes de modificarlo.

5. Matricular alumnos

Se debe comprobar que existe el Alumno y que no está matriculado de la asignatura.

6. Dar de baja de una asignatura

Se debe comprobar que existe el Alumno y que está matriculado de la asignatura.

7. Introducir calificación trimestral

Se debe comprobar que existe el Alumno.

8. Listar calificaciones de alumnos

Se debe comprobar que existe el Alumno.

9. Poner una falta (día completo)

Se debe comprobar que existe el Alumno.

10. Poner una falta (en una sesión)

Se debe comprobar que existe el Alumno.

11. Pasar lista

Se debe comprobar si hay alumnos.

12. Listar faltas

Se debe comprobar que existe el Alumno.

E. CÓDIGO FACILITADO

Clase Fecha

```
// Sobrecarga del método equals de la clase Object
	public boolean equals(Object object) {
		boolean igual = false;

		if (object instanceof Fecha) {
			Fecha fecha = (Fecha) object;
			if (fecha.getDia() == this.getDia() && fecha.getMes()
== this.getMes() && fecha.getAgno() == this.getAgno())
				igual = true;
		}
		return igual;
	}
```

Clase DiaClase

```
// Sobrecarga del método equals de la clase Object
	public boolean equals(Object object){
		boolean igual = false;

		if(object instanceof DiaClase){
			DiaClase diaClase = (DiaClase) object;
			if(diaClase.getDia().equals(this.getDia()))
				igual = true;
		}
		return igual;
	}
```

Clase Calificacion

```
// Sobrecarga del método equals de la clase Object
	public boolean equals(Object object){
		boolean igual = false;

		if(object instanceof Calificacion){
			Calificacion calificacion = (Calificacion) object;

	if(calificacion.getAsignatura().equalsIgnoreCase(this.getAsignat
ura()))
				igual = true;
		}
		return igual;
	}
```

Clase Alumno

```
// Sobrecarga del método equals de la clase Object
	public boolean equals(Object object){
		boolean igual = false;

		if(object instanceof Alumno){
			Alumno alumno = (Alumno) object;
			if(alumno.getDni().equalsIgnoreCase(this.getDni()))
				igual = true;
		}
		return igual;	}
```