DEPARTAMENTO LENGUAJES Y CIENCIAS DE LA COMPUTACIÓN	**PROGRAMACIÓN ORIENTADA A OBJETOS** (Prueba realizada el 17 de junio de 2013)	
APELLIDOS, Nombre	TITULACIÓN Y GRUPO	
	MÁQUINA	

NOTAS PARA LA REALIZACIÓN DEL EJERCICIO:

- El ejercicio se almacenará en el directorio **C:\POO**. En caso de que no exista deberá crearse, y si ya existiese, deberá borrarse todo su contenido antes de comenzar.
- Al inicio del contenido de cada fichero deberá indicarse **el nombre del alumno, titulación, grupo y código del equipo**que está utilizando**.**
- La evaluación tendrá en cuenta la claridad de los algoritmos, del código y la correcta elección de las estructuras de datos, así como los criterios de diseño que favorezcan la reutilización.
- Los diferentes apartados tienen una determinada puntuación. Si un apartado no se sabe hacer, el alumno no debe pararse en él indefinidamente. Puede abordar otros.
- **Está permitido:**
  - Consultar el API.
- **No está permitido**:
  - Utilizar otra documentación electrónica o impresa.
  - Intercambiar documentación con otros compañeros.
  - Utilizar soportes de almacenamiento.
- Una vez terminado el ejercicio subir un fichero comprimido (.rar) sólo con los **fuentes**

Se desea llevar a cabo la gestión de las prácticas de laboratorio realizadas por los alumnos de un determinado curso de programación. Para ello, se creará un proyecto `prEvaluacionPracticas` con las clases siguientes:

1) (***0.25ptos.***) La clase `EvaluacionException` se utilizará para tratar diferentes situaciones excepcionales tal y como se indica más adelante. Se trata de una excepción no comprobada.

2) (***1.25ptos.***) La clase `Alumno` debe mantener información sobre un alumno. En concreto, su dni (`String`), su nombre (`String`) y su calificación (`double`) obtenida por el alumno después de llevarse a cabo la evaluación de sus prácticas (en las clases `Evaluacion` y `EvaluacionRestringida`). También dispondrá del constructor y los métodos necesarios para:
   a. *(0.25ptos.)* Construir un objeto de la clase a partir del dni y el nombre del alumno (su calificación será 0). La representación de un alumno (`String toString()`) viene dada por esos tres datos, separados por espacios en blanco.
   b. *(0.25ptos.)* Obtener el dni (`String getDni()`), obtener el nombre (`String getNombre()`), obtener la calificación (`double getCalificacion()`) y poner calificación (`void setCalificacion(double calif)`).
   c. *(0.25ptos.)* Dos objetos de la clase `Alumno` son iguales si coinciden sus dni y sus nombres. No se tendrán en cuenta las diferencias por mayúsculas o minúsculas.
   d. *(0.5ptos.)* Un objeto de la clase `Alumno` a1 es menor que otro a2 cuando el nombre de a1 es menor alfabéticamente (independientemente de minúsculas y mayúsculas) que el nombre de a2. En caso de que los nombres sean iguales, el alumno a1 es menor que el a2 cuando el dni de a1 es menor alfabéticamente (independientemente de minúsculas y mayúsculas) que el dni de a2.

3) (***1.25ptos.***) La clase `Practica` debe mantener información sobre el nombre de una práctica de laboratorio (`String`) y la puntuación conseguida por un alumno en la misma (`double`). También dispondrá del constructor y los métodos necesarios para:
   a. *(0.25ptos.)* Construir un objeto de la clase a partir del nombre y la puntuación de la práctica. La representación de una práctica (`String toString()`) viene dada por el nombre.
   b. *(0.25ptos.)* Obtener el nombre (`String getNombre()`), obtener y establecer la puntuación (`int getPuntuacion() y void setPuntuacion(int Punt)`).
   c. *(0.25ptos.)* Dos objetos de la clase `Practica` son iguales si coinciden sus nombres (independientemente de que estén en mayúsculas o minúsculas).
   d. *(0.5ptos.)* Un objeto de la clase `Practica` p1 es menor que otro p2 cuando el nombre de p1 es menor alfabéticamente (con independencia de que estén en mayúsculas o minúsculas) que el de p2.

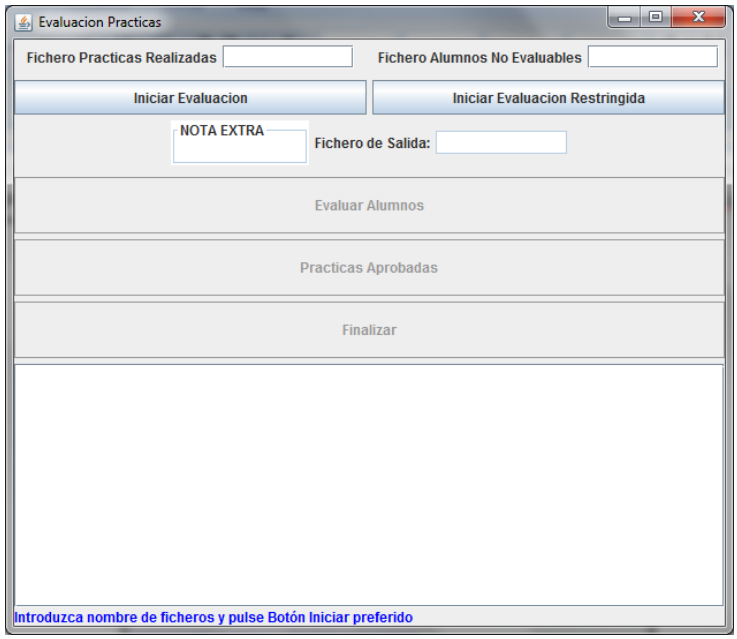
4) (**4.25*ptos.***) La clase `Evaluacion` almacenará la información de las prácticas y alumnos en una estructura de datos denominada `practicas` que asocia a cada alumno un conjunto ordenado de prácticas. Los alumnos estarán también ordenados en esta estructura por su orden natural. La clase dispondrá del constructor y los métodos necesarios para:

   a. *(1.5ptos.)* Construir un objeto de la clase a partir del nombre del fichero (`String`) que contiene la información necesaria para crear la estructura de datos `practicas`. El formato de los datos es el que aparece en el fichero `practicas.txt` proporcionado en el campus virtual. Cualquier error de formato detectado (falta algún dato de práctica o alumno, o bien el dato no es del tipo correcto), provocará el lanzamiento de una excepción del tipo `EvaluacionException`.
   b. *(1pto.)* Evaluar los alumnos en función de las prácticas realizadas (`void evaluarAlumnos(double extra)`). La calificación de un alumno se obtiene sumando las puntuaciones de las prácticas que ha realizado. Además, si el número de prácticas realizadas por este alumno supera o es igual a la media del número de prácticas realizadas por todos los alumnos, su calificación se verá incrementada con el valor recibido como parámetro (`extra`). Por el contrario, si el número de prácticas realizadas es menor que la media, dicho valor se le restará a la calificación obtenida.
   c. *(1pto.)* Calcular el número de alumnos que han superado (puntuación igual o superior a `5.0`) cada una de las prácticas. Aquellas prácticas en las que nadie aprobó no deben aparecer en la estructura devuelta (`SortedMap<Practica,Integer> practicasAprobadas()`).
   d. *(0.25ptos.)* Representar mediante una cadena de caracteres los alumnos almacenados en la estructura `practicas` (`String representarAlumnos()`). Cada alumno se representará en una línea.
   e. *(0.5ptos.)* Mostrar un listado de alumnos. Para ello se diseñarán dos métodos. El primero recibe el nombre del fichero (`void listarAlumnos(String nombreFichero)`) sobre el que se escribirá el listado. El segundo (`void listarAlumnos(PrintWriter pw)`) manda los datos por el objeto `PrintWriter` que recibe como parámetro. El formato de salida es como el mostrado en el fichero `calificaciones.txt` proporcionado en el campus virtual.

5) (***1pto.***) La clase `EvaluacionRestringida se comporta` como la clase `Evaluacion`, con la diferencia de que a la hora de evaluar a los alumnos no se tendrán en cuenta a aquellos que hayan tenido un número determinado de faltas de asistencia. Para ello, esta clase almacenará en la lista `alumnosNoEvaluables` (de tipo `List<Alumno>`) a dichos alumnos. Estos alumnos se leerán de un fichero. Dispondrá del constructor y el método necesarios para:

a. *(0.5ptos.)* Construir un objeto de la clase a partir del nombre del fichero (`String`) que contiene la información necesaria para crear la estructura de datos `practicas` y el nombre del fichero (`String`) que contiene los alumnos no evaluables (el formato de los datos es el que aparece en el fichero `noevaluables.txt` proporcionado en el campus virtual). La falta de algún dato de alumno provocará el lanzamiento de una excepción del tipo `EvaluacionException`.
b. *(0.5ptos.)* Una redefinición del método `void evaluarAlumnos(double extra)` para no considerar aquellos alumnos que estén en la lista `alummnosNoEvaluables`. Estos alumnos tendrán por defecto una calificación de 0.

6) (***2ptos.***) La clase `ControladorEvaluacion` controla e interactúa con el modelo (compuesto por las clases anteriormente descritas) y la vista (se proporcionan en el campus virtual la interfaz `VistaEvaluacion` y la clase `PanelEvaluacion`). El constructor debe habilitar la parte de inicialización de la vista (introducción de ficheros de prácticas y alumnos no evaluables, y los botones de iniciar evaluación) y mostrar un mensaje en la parte baja de la misma indicando al usuario que introduzca los nombres de los ficheros y pulse el botón iniciar deseado. El resto de la vista estará deshabilitado. La pulsación de alguno de los botones de iniciación hará que se cree un objeto de la clase `Evaluacion` o de la clase `EvaluacionRestringida`, respectivamente, se deshabilite la zona de inicialización, se habilite el resto de la vista y se muestre un mensaje al usuario invitándole a evaluar alumnos y a calcular prácticas aprobadas. Cada vez que el usuario pulse el botón “Evaluar Alumnos” se procederá a evaluar a los alumnos. El resultado se mostrará en el área de texto y se escribirá en el fichero de salida especificado. Cuando el usuario pulse el botón "Practicas Aprobadas" se calcularán las prácticas que han sido aprobadas por todos los alumnos que las han realizado y se mostrará el resultado en el área de texto. El botón “Finalizar” deshabilita esta zona de proceso, limpia las zonas de texto y habilita la parte de inicialización de la vista para poder empezar de nuevo con otros ficheros de entrada. Si al pulsar un determinado botón no se han introducido los datos necesarios en los campos de texto, se debe mostrar un mensaje de error en la parte baja de la vista (para ello se deben capturar excepciones de tipo `IOException` y `RuntimeException`)



La siguiente clase principal muestra un ejemplo del uso de las clases anteriormente descritas (SIN USO DE GUIs). Está disponible en el campus virtual.

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

public class Principal {

	public static void main(String[] args) {

		try {
			Evaluacion e = new Evaluacion("practicas.txt");

			e.evaluarAlumnos(5);

			// para presentar por pantalla
			System.out.println("Calificaciones de alumnos (Evaluacion Normal):\n");
			PrintWriter pw = new PrintWriter(System.out,true);
			e.listarAlumnos(pw);

			// para guardar en fichero
			e.listarAlumnos("calificaciones.txt");

			SortedMap<Practica,Integer> aprobadas = e.practicasAprobadas();

			System.out.println("\nPracticas aprobadas:\n");
			System.out.println(aprobadas.toString());

			// ahora EvaluacionRestringida
			EvaluacionRestringida er =
				new EvaluacionRestringida("practicas.txt","noevaluables.txt");

			er.evaluarAlumnos(5);

			// para presentar por pantalla
			System.out.println("\nCalificaciones de alumnos (EvaluacionRestringida):\n");
			PrintWriter pwr = new PrintWriter(System.out,true);
			er.listarAlumnos(pwr);

			// para guardar en fichero
			er.listarAlumnos("calificacionesRestringidas.txt");

		} catch (EvaluacionException e) {
			System.out.println("ERROR: " + e.getMessage());
		} catch (IOException e) {
			System.out.println("ERROR de Entrada/Salida: " + e.getMessage());
		}

	}

}
```

La siguiente clase principal hace uso de GUIs. Está disponible en el campus virtual.

```
import javax.swing.JFrame;
import javax.swing.JPanel;

public class PrincipalGUI {

	public static void main(String[] args) {

		VistaEvaluacion panel = new PanelEvaluacion();
		ControladorEvaluacion ctr = new ControladorEvaluacion(panel);
					// el modelo se crea dentro del controlador
		panel.controlador(ctr);

		JFrame ventana = new JFrame("Evaluacion Practicas");
		ventana.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
		ventana.setContentPane((JPanel) panel);
		ventana.pack();
		ventana.setVisible(true);
	}

}
```