DEPARTAMENTO LENGUAJES Y CIENCIAS DE LA COMPUTACIÓN	**PROGRAMACIÓN ORIENTADA A OBJETOS** (Prueba realizada el 17 de Septiembre de 2015)	
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 sólo con los ficheros **.java** que hayáis realizado a la tarea creada en el campus virtual para ello.

Se desea desarrollar una aplicación para la gestión de la asignación de asignaturas a profesores en un departamento universitario para un curso académico. Para ello, se creará un proyecto `prDocencia` con las clases siguientes:

1) (0.25 ***ptos.***) La clase `DocenciaException` se utilizará para tratar las diferentes situaciones excepcionales. Se trata de una excepción *no comprobada*. En general, se lanzará esta excepción si los parámetros de entrada de los métodos no contienen valores adecuados.
2) (***1.25 ptos.***) La clase `Docente` mantendrá información sobre un profesor del departamento. En concreto, su nombre (`String`), su capacidad docente o número de horas que debe impartir en un curso (`int`), su carga docente o número de horas que realmente imparte en el curso (`int`), y su horquilla (`int`), que se calculará como la diferencia entre la carga y la capacidad. Si la horquilla es negativa significará que imparte menos horas de las que debe y si es positiva más. La clase dispondrá de constructores y métodos necesarios para:
   a. Construir un objeto de la clase dados los valores para los dos primeros datos descritos anteriormente. Si la capacidad proporcionada no es un número positivo mayor que cero, se debe lanzar una excepción de tipo `DocenciaException` informando de tal situación. Inicializar la carga al valor 0 y asignar a la horquilla la diferencia entre la carga y la capacidad.
   b. Obtener el nombre (`String getNombre()`), la carga (`int getCarga()`), la capacidad (`int getCapacidad()`) y la horquilla (`int getHorquilla()`) . `void setCarga (int)` actualiza la carga del docente al valor pasado como parámetro, así como su correspondiente horquilla.
   c. La representación de un docente (`String toString()`) viene dada por su nombre, capacidad, carga y horquilla separados por el carácter ':'.
      Ej: Valeria Diaz:120:110:-10
   d. Dos objetos de la clase `Docente` son iguales si coinciden sus nombres, ignorando mayúsculas y minúsculas.
   e. Los objetos de la clase `Docente` se ordenan de forma natural por el nombre, ignorando mayúsculas y minúsculas.
3) (***1 pto.***) La clase `Asignatura` mantendrá información sobre una asignatura. En concreto, su nombre (`String`), su número de créditos expresados en horas (`int`), su código (`int`), y el código del centro donde se imparte (`int`). La clase dispondrá de constructores y métodos necesarios para:

a. Construir un objeto de la clase dados los valores para las cuatro variables descritas anteriormente. Si el número de horas o créditos de la asignatura es menor o igual que cero, se debe lanzar una excepción de tipo `DocenciaException` informando de tal situación.
b. Obtener el nombre (`String getNombre()`), el número de créditos (`int getCreditos()`), el código de la asignatura (`int getCodigoAsig()`) y el código del centro (`int getCodigoCentro()`).
c. La representación de una asignatura (`String toString()`) viene dada por su nombre, créditos, código de la asignatura y código del centro separados por el carácter ':'.

```
Sistemas Inteligentes:41:101065:101
```

d. Dos objetos de la clase `Asignatura` son iguales si coinciden sus códigos y sus números de créditos.
e. Los objetos de la clase `Asignatura` se ordenan de forma natural por el número de créditos (es menor el que menor número de créditos tenga) y en caso de igualdad por el código de la asignatura.

4) (***4 ptos.***) La clase `Asignacion` almacenará la información relativa a la asignación docente de un departamento en una aplicación ordenada denominada `asignacion`, que empareje a cada docente (objeto `Docente`) con el conjunto (también ordenado) de asignaturas (objetos de tipo `Asignatura`) que imparta dicho docente (`protected Map<Docente, Set <Asignatura>>`). La clase debe proporcionar los siguientes constructores y métodos públicos:

a. El constructor `Asignacion()` crea la estructura de datos vacía. Tantos los docentes como las asignaturas asociadas a cada docente se ordenarán por su orden natural.
El constructor `Asignacion(String)` recibe un nombre de fichero y, tras crear la estructura de datos, añade toda la información que contiene el fichero (ver apartados siguientes).
b. Los métodos protegidos `leeAsignacionFichero (String)` y `leeAsignacion(Scanner)` añaden información sobre la asignación a la estructura de datos. El primero lee la información de un fichero cuyo nombre se proporciona como parámetro. El segundo desde el flujo de entrada proporcionado. Se supondrá que la información sobre cada docente y sus asignaturas se encuentra en una línea de texto, donde el carácter `'&'` separa el docente de sus asignaturas y las distintas asignaturas entre sí. Para cada persona y cada asignatura sus datos se separan con el carácter `';'`. Por ejemplo, la siguiente línea indica que la docente Valeria Diaz tiene una capacidad de 120 y tiene asignadas dos asignaturas, Sistemas Inteligentes y Metodos Numericos. La primera tiene 60 horas, el código de la asignatura es 10167 y el código del centro 101.

```
Valeria Diaz;120&Sistemas Inteligentes;60;10167;101&Metodos Numericos;50;10145;101
```

Cualquier error de formato detectado (falta algún dato personal, o bien el dato no es del tipo correcto), provocará el lanzamiento de una excepción del tipo `DocenciaException`.

Se proporciona el fichero de datos `asigna.txt` para probar el correcto funcionamiento de la solución implementada. Modulariza la lectura de fichero usando los métodos `Docente leerDocente(String)` y `Asignatura leerAsignatura(String)` (este método será `protected`) a los que se les pasa las cadenas leídas de fichero con los datos del docente y los datos de cada asignatura respectivamente. Observa que para registrar al docente y al conjunto de asignaturas que imparte en la aplicación puedes usar el método del apartado c.

c. El método `insertaAsignacionProfesor(Docente, Set<Asignatura>)` registra al docente junto con las asignaturas que imparte en la estructura. Si el docente ya estuviese en la aplicación se añadirán las asignaturas indicadas en el segundo parámetro al conjunto de asignaturas que ya tuviera previamente asignadas. Calcula en ambos casos la carga docente de `Docente` sumando los créditos de las asignaturas que se le asignan, y establece con este valor la carga y la horquilla del profesor.
d. Se ha de redefinir el método `String toString()`. La representación de los objetos de la clase muestra toda la información contenida en la estructura de datos. El formato que se ha de utilizar es el que aparece en ejemplo de ejecución que os proporcionamos al final del enunciado.
e. Los métodos públicos `escribirFichero(String)` y `escribir(PrintWriter)` guardan la información de la estructura de datos en el fichero o en el flujo de salida dado respectivamente.
f. El método `double mediaAsignaturas()` calcula el número medio de asignaturas impartidas por los docentes.

g. El método `Set<Asignatura> encontrarDocencia(String)` devuelve el conjunto de asignaturas que imparte el docente cuyo nombre se pasa como parámetro. En caso de no encontrarse, se debe lanzar una excepción de tipo `DocenciaException` informando de tal situación.

5) (***2 ptos.***) La clase `AsignacionExtra` se **comporta como la clase** `Asignacion`, con la diferencia de que a la hora de asignar la docencia a los profesores, ha de añadirles una docencia extra de un determinado centro, en caso de que la horquilla de dicho docente sea negativa. Para ello la clase tendrá una variable de instancia `codigoCentro` con el código del centro y una lista denominada `docenciaSinAsignar (List <Asignatura>)` con una lista de asignaturas sin asignar. Para cada docente cuya horquilla sea negativa se seleccionará la primera asignatura de `docenciaSinAsignar` cuyo código de centro coincida con el valor de `codigoCentro` y la añadirá al conjunto de las asignaturas del profesor. Además, se actualizará la carga y la horquilla de dicho docente.

a. El constructor `AsignacionExtra(int)` crea la lista vacía y asigna el valor recibido como parámetro a la variable `codigoCentro`. El constructor `AsignacionExtra(int, String, String),` además de lo anterior, añadirá a la lista `docenciaSinAsignar` todas las asignaturas almacenadas en el fichero cuyo nombre se recibe en el segundo parámetro. Para ello se implementa el método `leeAsignaturasFichero (String)`. Dicho fichero almacenará la información de cada asignatura en una línea con el formato siguiente (se proporciona como ejemplo "asignaturas.txt"):

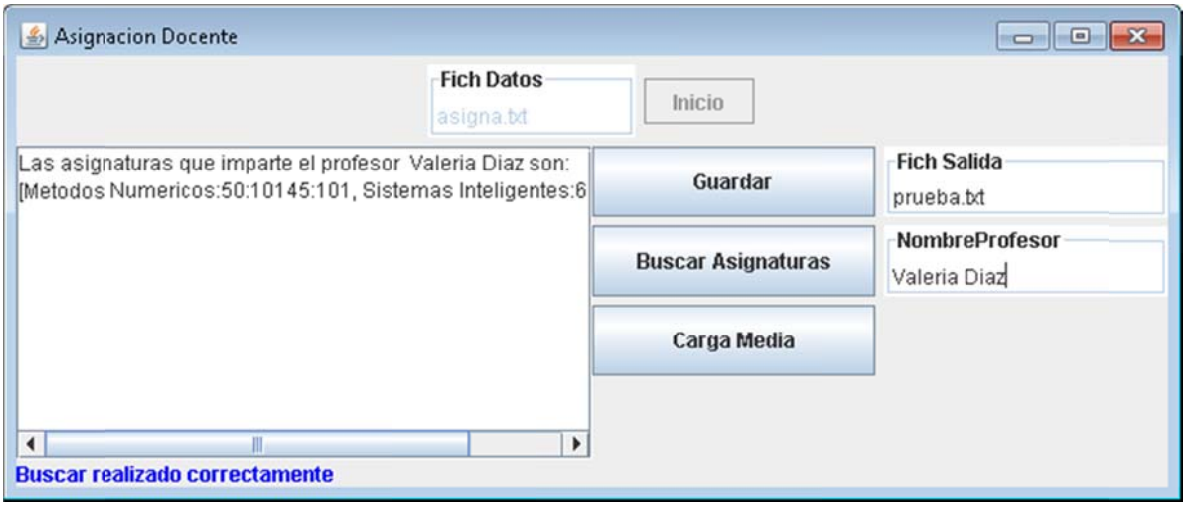
```
Fundamentos de Informática;60;102178;102
```

Observa que para implementar el método `leeAsignaturasFichero` puedes reutilizar el método `leerAsignatura` implementado en la clase `Asignacion.` A continuación se añadirá toda la información del fichero cuyo nombre se pasa como tercer parámetro. El formato de dicho fichero es el mismo que el utilizado para la clase `Asignacion` (se proporciona como ejemplo asigna.txt).

b. Redefinir `insertaAsignacionProfesor(Docente, Set<Asignatura>).` La nueva versión del método además de añadir a la estructura de datos el conjunto de asignaturas al docente, ha de añadir a éste una asignatura extra si su horquilla es negativa. En este caso seleccionará de la lista `docenciaSinAsignar` la primera asignatura cuyo código coincida con el valor de `codigoCentro`. Además, dicha asignatura será eliminada de la lista `docenciaSinAsignar`. En caso de que no haya ninguna con ese código se elevará una excepción del tipo `DocenciaException`. Además se habrá de actualizar la carga y la horquilla del docente teniendo en cuenta la asignatura extra asignada.

c. Redefinir el método `String toString()` para que a la información sobre los docentes y sus asignaturas asignadas mostrada en la clase `Asignación`, añada todos los asignaturas que queden sin asignar en la lista `docenciaSinAsignar`.

6) (***1.5 ptos.***) La clase `ControladorDocencia` controla e interactúa con el modelo (clases `Asignacion` y `Docente`) y la vista (se proporcionan en el campus virtual la interfaz `VistaDocencia` y la clase `PanelDocencia`). El constructor debe habilitar la parte de inicialización de la vista (introducción del fichero de profesores y el botón de inicio) y mostrar un mensaje en la parte baja de la misma indicando al usuario que introduzca el nombre del fichero y pulse el botón iniciar. El resto de la vista estará deshabilitado. La pulsación del botón "Inicio" hará que se cree un objeto de la clase `Asignacion` (pasándole el nombre del fichero introducido), se deshabilite la zona de inicialización y se habilite el resto de la vista. Cada vez que el usuario pulse algunos de los demás botones se procederá a realizar la acción correspondiente. El botón "Guardar" guarda la información sobre docentes en el fichero indicado en el campo de texto a su derecha. En caso de que dicho campo contenga la cadena vacía, la información se imprime en el área de texto del panel. . El botón "Buscar Asignaturas" hará que se lea el nombre del profesor del campo de texto de su derecha y se muestre en el área de texto los datos de todas las asignaturas que imparte dicho profesor. El botón "Carga Media" muestra en el área de texto el resultado de calcular el número medio de asignaturas impartidas por un profesor. Tras cada operación se debe mostrar un mensaje de confirmación o error en la parte baja de la vista.



Las clases `Main` y `MainGUI` se proporcionan en el campus virtual, para que se pueda probar el funcionamiento de las distintas clases a implementar. A continuación tenéis la salida que se ha de obtener en consola tras la ejecución del mismo con los ficheros proporcionados.

```
Datos leidos:
Pablo Lopez:84:102:18&Calculo:18:10199:101&Estructuras de Datos:41:10178:101&Complejidad y
Calculabilidad:43:10167:101
Valeria Diaz:120:110:-10&Metodos Numericos:50:10145:101&Sistemas Inteligentes:60:10167:101
Violeta Escudero:52:41:-11&Programacion Orientada a Objetos:41:10167:101

Las asignaturas que imparte el profesor Pablo Lopez son: [Calculo:18:10199:101, Estructuras de
Datos:41:10178:101, Complejidad y Calculabilidad:43:10167:101]
La media de asignaturas impartidas por profesor es: 2.0
Enviando la información a la salida estándar
Pablo Lopez:84:102:18&Calculo:18:10199:101&Estructuras de Datos:41:10178:101&Complejidad y
Calculabilidad:43:10167:101
Valeria Diaz:120:110:-10&Metodos Numericos:50:10145:101&Sistemas Inteligentes:60:10167:101
Violeta Escudero:52:41:-11&Programacion Orientada a Objetos:41:10167:101
Guardar Datos en salida.txt

Probando Asignacion Extra
Datos leidos:
Pablo Lopez:84:102:18&Calculo:18:10199:101&Estructuras de Datos:41:10178:101&Complejidad y
Calculabilidad:43:10167:101
Valeria Diaz:120:140:20&Programacion Orientada de Videojuegos:30:10169:101&Metodos
Numericos:50:10145:101&Sistemas Inteligentes:60:10167:101
Violeta Escudero:52:89:37&Programacion Orientada a Objetos:41:10167:101&Analisis y Diseño de
Algoritmos:48:10161:101

Fundamentos de Informática:60:102178:102
Metodos Estadisticos:50:10976:109
```

DEPARTAMENTO
LENGUAJES Y CIENCIAS DE
LA COMPUTACIÓN

PROGRAMACIÓN ORIENTADA A OBJETOS
(Prueba realizada el 17 de Septiembre de 2015)

APELLIDOS, Nombre	TITULACIÓN Y GRUPO	
	MÁQUINA