APELLIDOS, Nombre	TITULACIÓN Y GRUPO	
	MÁQUINA	

NOTAS PARA LA REALIZACIÓN DEL EJERCICIO:

- Al inicio del contenido de cada fichero realizado deberá aparecer un comentario con tus apellidos y nombre, titulación y grupo.
- Los diferentes apartados tienen una determinada puntuación. Si un apartado no se sabe hacer, no debes pararte en él indefinidamente. Puedes abordar otros.
- Está permitido:
  - Consultar los apuntes (CV), la API (Internet), la guía rápida de la API (CV).
  - Añadir métodos privados a las clases.
- No está permitido:
  - Intercambiar documentación con otros compañeros.
  - Recibir ayuda de otras personas. Se debe realizar personal e individualmente la solución del ejercicio propuesto.
  - Añadir métodos no privados a las clases.
  - Añadir variables o constantes a las clases.
  - Modificar la visibilidad de las variables, constantes y métodos que aparecen en el diagrama UML.
  - Modificar el código suministrado.
- Una vez terminado el ejercicio, debéis subir (a la tarea creada en el campus virtual para ello) un archivo comprimido de la carpeta `src` del proyecto (src.zip).
- 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.
- Para la corrección del ejercicio se utilizarán **programas de detección de copias/plagios**.
- Con posterioridad a la realización del ejercicio, el profesor podrá convocar a determinado/as alumno/as para realizar **entrevistas personales síncronas** con objeto de comprobar la autoría de las soluciones entregadas.

Proyecto `prSept20` y Paquete `pruebas`

En este examen se va a desarrollar un conjunto de clases (ver diagrama de la figura 1) que nos permitirá calcular el porcentaje de éxito de las pruebas JUnit aplicadas a las prácticas de los alumnos. Para ello, se creará un proyecto `prSept20` con las clases siguientes en el paquete `pruebas`. Todas las variables de instancia son privadas.

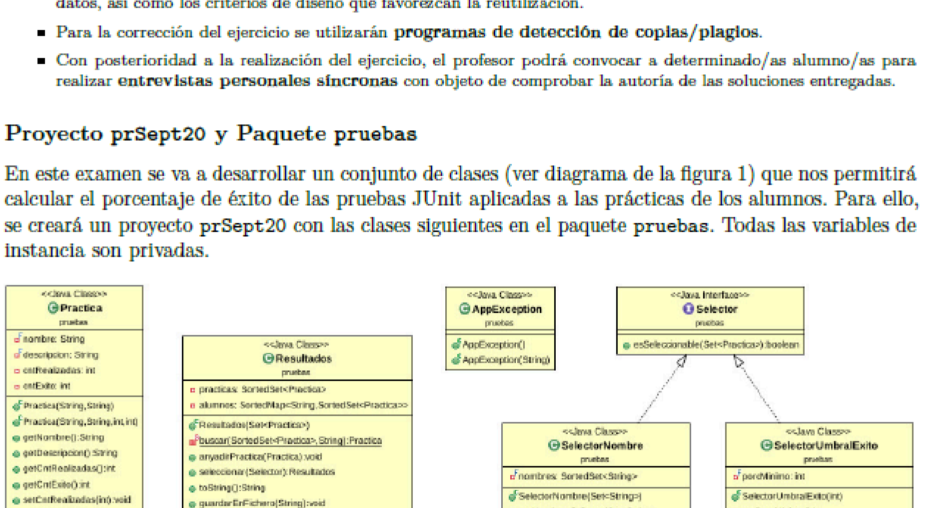

Figura 1: Diagrama de clases

La excepción `AppException` (0.25 pts.)

Defina la clase `AppException` (del paquete `pruebas`) como una **excepción no comprobada** para notificar las situaciones excepcionales que se produzcan.

La clase `Practica` (1.50 pts.)

La clase `Practica` (del paquete `pruebas`) contiene información sobre el resultado de aplicar un determinado número de pruebas JUnit a una práctica de un alumno, tal como el nombre del alumno, la descripción de la práctica, la cantidad de pruebas realizadas, y la cantidad de pruebas superadas con éxito.

- `Practica(String,String)`
  El constructor recibe, en el siguiente orden, el nombre del alumno y la descripción de la práctica. Si los parámetros son erróneos (nombre o descripción *nulo* o de *longitud cero*), entonces lanza una excepción de tipo `AppException`. En otro caso almacena los valores recibidos e inicializa la cantidad de pruebas realizadas y la cantidad de pruebas superadas con éxito a cero.
- `Practica(String,String,int,int)`
  El constructor recibe, en el siguiente orden, el nombre del alumno, la descripción de la práctica, la cantidad de pruebas realizadas y la cantidad de pruebas superadas con éxito. Si los parámetros son erróneos (nombre o descripción *nulo* o de *longitud cero*, o algún número negativo), entonces lanza una excepción de tipo `AppException`. En otro caso almacena los valores recibidos.
- `getNombre(): String`
  Devuelve el nombre del alumno.
- `getDescripcion(): String`
  Devuelve la descripción de la práctica.
- `getCntRealizadas(): int`
  Devuelve la cantidad de pruebas realizadas.
- `getCntExito(): int`
  Devuelve la cantidad de pruebas superadas con éxito.
- `setCntRealizadas(int): void`
  Si el valor recibido como parámetro es negativo, entonces lanza una excepción de tipo `AppException`. En otro caso, asigna el valor recibido como parámetro a la cantidad de pruebas realizadas.
- `setCntExito(int): void`
  Si el valor recibido como parámetro es negativo, entonces lanza una excepción de tipo `AppException`. En otro caso, asigna el valor recibido como parámetro a la cantidad de pruebas superadas con éxito.
- `getPorcExito(): int`
  Si el valor de la cantidad de pruebas realizadas es cero, entonces devuelve el valor cero. En otro caso, devuelve el porcentaje de éxito de la práctica, un número entero entre 0 y 100, ambos inclusive, como el porcentaje de la cantidad de pruebas superadas con éxito con respecto a la cantidad de pruebas realizadas: $\frac{100 \times cntExito}{cntRealizadas}$
- `toString(): String`
  Devuelve la representación textual del objeto, en el formato del siguiente ejemplo (incluyendo los paréntesis), donde el valor 30 representa la cantidad de pruebas realizadas, el valor 15 representa la cantidad de pruebas superadas con éxito, y 50 % representa el porcentaje de éxito de esta práctica:

  ```
  (nombre, descripción, 30, 15, 50%)
  ```
- Se considera que dos objetos de la clase `Practica` son **iguales** si el nombre del alumno y la descripción de la práctica son iguales en ambos objetos, sin diferenciar mayúsculas de minúsculas en ambas comparaciones.
- Los objetos de la clase `Practica` proporcionan el **orden natural** comparando el nombre del alumno, y en caso de igualdad, entonces se compara la descripción de la práctica de ambos objetos, sin diferenciar mayúsculas de minúsculas en ambas comparaciones.

La clase `MainPractica` (0.50 pts.)

Desarrolle la aplicación (clase distinguida en el paquete anónimo) `MainPractica` que permita comprobar los aspectos fundamentales de la clase `Practica`. Para ello, la aplicación debe crear los siguientes objetos de la clase `Practica` con los siguientes valores:

- *práctica-1*: `("Pepe", "pr1", 20, 15)`
- *práctica-2*: `("pepe", "PR1", 22, 10)`
- *práctica-3*: `("paco", "pr3", 25, 12)`
- *práctica-4*: `("paco", "pr4", 30, 17)`

A continuación, comprobará si *práctica-1* es igual a *práctica-2*, en caso afirmativo, mostrará en consola el mensaje `"iguales"`, en otro caso, mostrará en consola el mensaje `"distintas"`. Además, comprobará si *práctica-2* es igual a *práctica-3*, en caso afirmativo, mostrará en consola el mensaje `"iguales"`, en otro caso, mostrará en consola el mensaje `"distintas"`. Finalmente, comprobará si *práctica-3* es igual a *práctica-4*, en caso afirmativo, mostrará en consola el mensaje `"iguales"`, en otro caso, mostrará en consola el mensaje `"distintas"`.

A continuación, se debe asignar el valor 18 a la cantidad de pruebas superadas con éxito de *práctica-1*. Además, se debe asignar el valor 40 a la cantidad de pruebas realizadas de *práctica-3*.

Por último, se debe crear un *conjunto ordenado de prácticas*, se deben añadir, en el orden especificado, los cuatro objetos anteriores a dicho conjunto, y finalmente se mostrará en consola la representación textual del conjunto.

La ejecución de esta aplicación debería mostrar en consola la siguiente información:

```
iguales
distintas
distintas
[(paco, pr3, 40, 12, 30%), (paco, pr4, 30, 17, 56%), (Pepe, pr1, 20, 18, 90%)]
```

La interfaz `Selector` (0.25 pts.)

La interfaz `Selector` (del paquete `pruebas`) especifica un método que permite seleccionar un determinado conjunto de prácticas dependiendo de diversas circunstancias, determinadas por las clases que implementen esta interfaz.

- `esSeleccionable(Set<Practica>): boolean`
  Devuelve `true` si el conjunto de prácticas recibido como parámetro debe ser seleccionado por este selector. En otro caso devuelve `false`.

La clase `SelectorNombre` (0.75 pts.)

La clase `SelectorNombre` (del paquete `pruebas`) permite seleccionar un determinado conjunto de prácticas si el nombre del alumno vinculado a dichas prácticas se encuentra, sin diferenciar letras mayúsculas de minúsculas, en el conjunto de nombres especificado durante la construcción del objeto.

Esta clase implementa la interfaz `Selector`, y contiene información sobre los nombres de los alumnos que serán seleccionados.

- `SelectorNombre(Set<String>)`
  Construye un nuevo objeto y almacena en un conjunto ordenado los nombres contenidos en el conjunto recibido como parámetro, considerando que si el conjunto recibido como parámetro es `null`, entonces lanza una excepción de tipo `AppException`. Además los nombres se deben almacenar con su representación en **mayúsculas**.
- `getNombres(): SortedSet<String>`
  Devuelve el conjunto ordenado con los nombres en mayúsculas especificados durante la construcción de este objeto.
- `esSeleccionable(Set<Practica>): boolean`
  Este método recibe como parámetro un conjunto de prácticas, todas del mismo alumno (no es necesario comprobarlo). Si el conjunto está vacío, entonces devuelve `false`. En otro caso, devuelve `true` si el nombre del alumno vinculado a las prácticas del conjunto recibido como parámetro se encuentra (sin diferenciar letras mayúsculas de minúsculas) en el conjunto de nombres especificado durante la construcción de este objeto. En cualquier otro caso devuelve `false`.

La clase `SelectorUmbralExito` (0.75 pts.)

La clase `SelectorUmbralExito` (del paquete `pruebas`) implementa la interfaz `Selector`, y contiene información sobre el porcentaje de éxito mínimo del total de las prácticas que un alumno debe superar para que sus prácticas que sean seleccionadas.

Permite seleccionar las prácticas de un alumno si el porcentaje de éxito del total de prácticas del alumno es mayor o igual a un determinado porcentaje mínimo, especificado durante la construcción del objeto.

- `SelectorUmbralExito(int)`
  Construye un nuevo objeto y almacena el valor recibido como parámetro como porcentaje de éxito mínimo del total de prácticas requerido para seleccionar las prácticas de un alumno, considerando que si el valor recibido como parámetro es negativo, entonces lanza una excepción de tipo `AppException`.
- `getPorcMinimo(): int`
  Devuelve el valor del porcentaje de éxito mínimo del total de prácticas, especificado durante la construcción de este objeto.
- `esSeleccionable(Set<Practica>): boolean`
  Este método recibe como parámetro un conjunto de prácticas, todas del mismo alumno (no es necesario comprobarlo). Debe calcular el valor del porcentaje de éxito total de todas las prácticas del conjunto de prácticas recibido como parámetro, según la siguiente fórmula $\frac{100 \times \sum cntExito}{\sum cntRealizadas}$
  Devuelve `true` si el porcentaje de éxito total calculado es mayor o igual al porcentaje de éxito mínimo del total de prácticas especificado durante la construcción de este objeto. En otro caso devuelve `false`.

La clase `Resultados` (6.00 pts.)

La clase `Resultados` (del paquete `pruebas`) contiene información sobre los resultados de las pruebas JUnit aplicadas a las prácticas de los alumnos.

Esta clase posee una variable de instancia, denominada `practicas`, que es un conjunto ordenado con información sobre las prácticas que se han diseñado para los alumnos. Las prácticas de este conjunto sólo almacenan, como información relevante, la descripción y la cantidad de pruebas realizadas (pruebas JUnit que se aplicarán para su corrección). Tanto el nombre del alumno como la cantidad de pruebas superadas con éxito de estas prácticas no es relevante, y podrán contener cualquier valor válido.

Esta clase tiene una segunda variable de instancia, denominada `alumnos`, que es una correspondencia ordenada que asocia el nombre de cada alumno con el conjunto ordenado de las prácticas que ha realizado.

- `Resultados(Set<Practica>)`
  Construye el objeto creando vacía la correspondencia **ordenada** de las prácticas de los alumnos, denominada `alumnos`. Además, crea un nuevo conjunto **ordenado** de prácticas, denominado `practicas`, y le añade todas las prácticas del conjunto recibido como parámetro.
- `buscar(SortedSet<Practica>,String): Practica`
  Es un método de **clase privado** que busca en el conjunto de prácticas recibido como primer parámetro, una práctica con **descripción** igual, sin diferenciar mayúsculas de minúsculas, a la cadena recibida como segundo parámetro. En caso de encontrarla, devuelve la práctica encontrada, en otro caso devuelve `null`.
- `anyadirPractica(Practica): void`
  Este método recibe como parámetro la práctica de un alumno y la añade al conjunto de prácticas de la correspondencia `alumnos` asociado a ese alumno.
  Antes de añadirla, hay que actualizar la cantidad de pruebas realizadas de esa práctica. Para ello, este método busca en el conjunto denominado `practicas`, una práctica con una descripción igual a la de la práctica recibida como parámetro, sin diferenciar mayúsculas de minúsculas. Si no la encuentra, lanza una excepción de tipo `AppException`. En otro caso, toma el valor de la cantidad de pruebas realizadas de la práctica encontrada, y lo asigna a la cantidad de pruebas realizadas de la práctica recibida como parámetro.
  Por ejemplo, si la práctica recibida como parámetro es *("Pablo", "pr2", 30, 20)*, y el conjunto `practicas` contiene las prácticas *("Practica", "pr1", 35, 0)*, *("Practica", "pr2", 45, 0)*, *("Practica", "pr3", 55, 0)*, entonces se actualizaría el valor de la cantidad de pruebas realizadas a 45, quedando *("Pablo", "pr2", 45, 20)* (nótese que el valor de la cantidad de pruebas superadas con éxito no se modifica).
  Una vez hecho esto, la práctica recibida como parámetro (con el valor actualizado de la cantidad de pruebas realizadas) se añadirá, dentro de la correspondencia `alumnos`, al conjunto asociado con el nombre del alumno (en minúsculas). Si ya hay alguna práctica igual a la recibida como parámetro, entonces la nueva práctica debe **reemplazar** a la que ya estuviese en dicho conjunto.
- `seleccionar(Selector): Resultados`
  Este método crea una instancia de la clase `Resultados` con los resultados de la ejecución de las pruebas para una determinada selección de alumnos. Aquellos alumnos cuyo conjunto prácticas satisfaga la condición especificada serán incluidos.
  Para ello, crea un nuevo objeto de la clase `Resultados` (con las mismas prácticas del conjunto `practicas` del objeto actual) al que añadirá (invocando a `anyadirPractica`) todas las prácticas de todos aquellos conjuntos de prácticas de la correspondencia `alumnos` del objeto actual, para los cuales la invocación al método `esSeleccionable` aplicado al objeto `Selector` recibido como parámetro devuelve `true`.
  En la invocación al método `esSeleccionable`, el conjunto de prácticas asociado a cada nombre de alumno será pasado como parámetro.
  Finalmente, este nuevo objeto de la clase `Resultados` con las prácticas seleccionadas será devuelto como resultado del método.
- `toString(): String`
  Devuelve la representación textual del objeto, según el formato del siguiente ejemplo, donde primero se representan las prácticas del conjunto `practicas`, y posteriormente las prácticas de la correspondencia `alumnos` (se han añadido saltos de línea para mejorar la legibilidad, pero no deben ser incluidas en el procesamiento):

  ```
  { [ (Practica, pr1, 35, 0, 0%); (Practica, pr2, 45, 0, 0%); (Practica, pr3, 55, 0, 0%) ],
    [ (ana luisa, pr2, 45, 35, 77%); (juan luis, pr2, 45, 12, 26%); (juan luis, pr3, 55, 30, 54%);
      (maria luisa, pr1, 35, 13, 37%); (maria luisa, pr2, 45, 20, 44%);
      (pepe luis, pr1, 35, 10, 28%); (pepe luis, pr3, 55, 13, 23%) ] }
  ```
- `guardarEnFichero(String): void`
  Guarda en el fichero, cuyo nombre se recibe como parámetro, todas las prácticas de la correspondencia `alumnos`. En caso de error al escribir en el fichero, propaga la excepción `IOException`.
  En el fichero, cada línea contiene los siguientes datos de una práctica (nombre, descripción y cantidad de pruebas superadas con éxito), separados por el símbolo punto y coma (;) seguido de espacio, según el formato del siguiente ejemplo (nótese que la cantidad de pruebas realizadas no se incluye):

  ```
  ana luisa; pr2; 35
  juan luis; pr2; 12
  juan luis; pr3; 30
  maria luisa; pr1; 13
  maria luisa; pr2; 20
  pepe luis; pr1; 10
  pepe luis; pr3; 13
  ```
- `cargarDeFichero(String): void`
  Lee del fichero, cuyo nombre se recibe como parámetro, los datos de las prácticas, y las añade a la correspondencia `alumnos` (utilizando para ello el método `anyadirPractica` especificado anteriormente). En caso de error de lectura del fichero, propaga la excepción `IOException`.

  En el fichero, cada línea contiene los siguientes datos de una práctica (nombre, descripción y cantidad de pruebas superadas con éxito), separados por los siguientes delimitadores: `"\\s*[;]\\s*"`, según el formato del siguiente ejemplo (nótese que la cantidad de pruebas realizadas no se incluye, y le podemos dar el valor cero, ya que será re-asignada automáticamente en la invocación al método `anyadirPractica`):

  ```
  maria luisa ; pr1 ; 9
  pepe luis ; pr3 ; 8
  ana luisa ; pr2 ; 7
  pepe luis ; pr1 ; 10
  juan luis ; pr2 ; 12
  maria luisa ; pr1 ; 13
  juan luis ; pr3 ; 30
  maria luisa ; pr2 ; 20
  pepe luis ; pr3 ; 13
  ana luisa ; pr2 ; 35
  ```
  Si existe algún error en el formato de los datos de entrada, se desecharán los datos de la línea involucrada, y se continuará con los datos de las siguientes líneas.
- `completarPracticas(): void`
  Para cada alumno de la correspondencia `alumnos`, comprueba si ha entregado todas las prácticas, según el conjunto `practicas`, y en caso de que no haya entregado todas las prácticas, entonces creará y añadirá (invocando al método `anyadirPractica`) todas aquellas prácticas con el nombre del alumno y las descripciones de las prácticas que no haya realizado, con ambas cantidades (`cntRealizadas` y `cntExito`) con el valor cero.
  Ayuda: téngase en cuenta que no se puede modificar una correspondencia ni un conjunto mientras se itera sobre ellos.
  Por ejemplo, dado el siguiente conjunto `practicas`:
  `[ (Practica, pr1, 35, 0, 0%), (Practica, pr2, 45, 0, 0%), (Practica, pr3, 55, 0, 0%) ]`
  A partir de las prácticas mostradas en la izquierda, calcula y añade las prácticas de los alumnos que no han sido realizadas (mostradas a la derecha):

(ana luisa; pr2; 45; 35; 77%)	(ana luisa; pr1; 35; 0; 0%) <- (NUEVO) (ana luisa; pr2; 45; 35; 77%) (ana luisa; pr3; 55; 0; 0%) <- (NUEVO)
(juan luis; pr2; 45; 12; 26%) (juan luis; pr3; 55; 30; 54%)	(juan luis; pr1; 35; 0; 0%) <- (NUEVO) (juan luis; pr2; 45; 12; 26%) (juan luis; pr3; 55; 30; 54%)
(maria luisa; pr1; 35; 13; 37%) (maria luisa; pr2; 45; 20; 44%)	(maria luisa; pr1; 35; 13; 37%) (maria luisa; pr2; 45; 20; 44%) (maria luisa; pr3; 55; 0; 0%) <- (NUEVO)
(pepe luis; pr1; 35; 10; 28%) (pepe luis; pr3; 55; 13; 23%)	(pepe luis; pr1; 35; 10; 28%) (pepe luis; pr2; 45; 0; 0%) <- (NUEVO) (pepe luis; pr3; 55; 13; 23%)

La clase `MainResultados`

Descargue del campus virtual el fichero de datos `"datos.txt"`, y cópielo a la carpeta base raíz del proyecto. Además, descargue el fichero `MainResultados.java` y cópielo al paquete anónimo (por defecto).

La aplicación (clase distinguida en el paquete anónimo) `MainResultados` permite comprobar los aspectos fundamentales de las clases anteriores.

La ejecución de esta aplicación debería mostrar en consola la siguiente información. Se han añadido saltos de línea para mejorar la legibilidad. Nótese que el mensaje `"Fin de ejecucion"` no es mostrado:

```
{ [ (Practica, pr1, 35, 0, 0%); (Practica, pr2, 45, 0, 0%); (Practica, pr3, 55, 0, 0%) ],
  [ (ana luisa, pr2, 45, 35, 77%); (juan luis, pr2, 45, 12, 26%); (juan luis, pr3, 55, 30, 54%);
    (maria luisa, pr1, 35, 13, 37%); (maria luisa, pr2, 45, 20, 44%);


    (pepe luis, pr1, 35, 10, 28%); (pepe luis, pr3, 55, 13, 23%) ] }
{ [ (Practica, pr1, 35, 0, 0%); (Practica, pr2, 45, 0, 0%); (Practica, pr3, 55, 0, 0%) ],
  [ (ana luisa, pr1, 35, 0, 0%); (ana luisa, pr2, 45, 35, 77%); (ana luisa, pr3, 55, 0, 0%);
    (juan luis, pr1, 35, 0, 0%); (juan luis, pr2, 45, 12, 26%); (juan luis, pr3, 55, 30, 54%);
    (maria luisa, pr1, 35, 13, 37%); (maria luisa, pr2, 45, 20, 44%); (maria luisa, pr3, 55, 0, 0%);
    (pepe luis, pr1, 35, 10, 28%); (pepe luis, pr2, 45, 0, 0%); (pepe luis, pr3, 55, 13, 23%) ] }
{ [ (Practica, pr1, 35, 0, 0%); (Practica, pr2, 45, 0, 0%); (Practica, pr3, 55, 0, 0%) ],
  [ (ana luisa, pr1, 35, 0, 0%); (ana luisa, pr2, 45, 35, 77%); (ana luisa, pr3, 55, 0, 0%);
    (maria luisa, pr1, 35, 13, 37%); (maria luisa, pr2, 45, 20, 44%); (maria luisa, pr3, 55, 0, 0%) ] }
{ [ (Practica, pr1, 35, 0, 0%); (Practica, pr2, 45, 0, 0%); (Practica, pr3, 55, 0, 0%) ],
  [ (ana luisa, pr1, 35, 0, 0%); (ana luisa, pr2, 45, 35, 77%); (ana luisa, pr3, 55, 0, 0%);
    (juan luis, pr1, 35, 0, 0%); (juan luis, pr2, 45, 12, 26%); (juan luis, pr3, 55, 30, 54%);
    (maria luisa, pr1, 35, 13, 37%); (maria luisa, pr2, 45, 20, 44%); (maria luisa, pr3, 55, 0, 0%) ] }

Error: Descripcion erronea xxx
```

La ejecución de esta aplicación también debería almacenar en el fichero denominado `"salida.txt"` (en la carpeta base raíz del proyecto) la siguiente información:

```
ana luisa; pr1; 0
ana luisa; pr2; 35
ana luisa; pr3; 0
juan luis; pr1; 0
juan luis; pr2; 12
juan luis; pr3; 30
maria luisa; pr1; 13
maria luisa; pr2; 20
maria luisa; pr3; 0
pepe luis; pr1; 10
pepe luis; pr2; 0
pepe luis; pr3; 13
```