CURSO 2021-22.
Práctica 2: Pilas y Colas.

1. OBJETIVOS:

- Utilizar estructuras de datos Pila para resolver diferentes algoritmos recursivos.
- Utilizar estructuras de datos Cola para resolver diferentes algoritmos iterativos.

2. Apertura Proyecto.

2.1. Abrir el proyecto "*ED 2 Practica. PilasColas".*

Descargue el archivo "ED 2 Practica. PilasColas.zip" de *Moodle* y descomprímalo. A continuación, abra el proyecto con *IntelliJ* y compruebe los archivos Java que contiene:

- **NodoPila**, **Pila, NodoCola** y **Cola** son las clases que implementan los TAD Pila y Cola a través de listas enlazadas, de forma similar a como se ha visto en clase.
- **Asignatura** y **Alumno** son clases vacías en las que se va a trabajar en esta práctica. Las clases **Principal**, **AlgoritmosPila** y **AlgoritmosCola** contienen código que es necesario completar.

Definición del JDK para el proyecto.

Es muy posible que al tratar de abrir un fichero Java del proyecto, aparezca en la parte superior de la pantalla el mensaje *"Project SDK is not defined",* además de errores en el código fuente indicando que no se reconocen las clases de biblioteca (*System*, *String*, etc). Esto es debido a que el proyecto ha sido creado con una versión de JDK diferente a la que tiene el sistema en donde va a hacer la práctica.

El problema se soluciona fácilmente, definiendo de forma adecuada el JDK para el proyecto. Se proponen a continuación dos métodos diferentes:

1. Junto al mensaje de error *"Project SDK is not defined"* aparece un enlace *"Setup SDK"*. Al pulsarlo, aparecen los JDK de los que tiene constancia *IntelliJ*. Se elige uno cualquiera y el problema queda resuelto. En caso de que no aparezca ninguno, se utilizará el segundo método, que se propone a continuación.
2. En la barra de menú de *IntelliJ*, pulsar ***File --> Project Structure***. En ***Project Settings***, ***Project***, existe una lista desplegable para ***Project SDK***. En ella aparecen los JDK de los que tiene constancia *IntelliJ*, se seleccionará uno cualquiera y el problema quedará resuelto.

   En caso de que *IntelliJ* no tenga constancia de ningún JDK, se pulsará el botón ***Edir*** para definir uno. Se selecciona la opción JDK, se le indica la ubicación del JDK en el sistema de archivos, por ejemplo, C:\Program Files\Java\jdk-11.0.5 y se pulsa OK.

Nota: En el programa principal se deberá sustituir el comentario de Nombre y Apellidos Alumnos, y el numero matricula, por los datos correspondientes del alumno/a.

Nota: Además de las pruebas propuestas, el alumno/a podrá realizar las pruebas adicionales que considere oportunas. Al realizar la entrega, únicamente hay que entregar el proyecto con las pruebas indicadas en el enunciado.

3. Algoritmos con Pilas.

A lo largo de esta parte de la práctica, se va a trabajar con una pila de asignaturas. La clase *Asignatura* contendrá dos atributos: *nombre* (de tipo *String*) y *calificación* (de tipo *double*).

1. Definir la clase ***Asignatura***.

   Cada objeto de la clase *Asignatura* contendrá encapsulados los datos o información de una asignatura en la que está matriculado un alumno: el nombre de la asignatura, y la calificación. Para almacenar estos datos, se decide utilizar los siguientes **atributos** en la clase *Asignatura*:

   - *nombre* de tipo *String*.
   - *calificacion*: de tipo *double.*

   La interfaz de esta clase, es decir, las operaciones que ofrece la clase *Asignatura*, vienen definidas por los siguientes **métodos**:

   - Constructor con dos parámetros, inicializando los atributos con los valores recibidos.
   - Método que devuelve el nombre de la asignatura: *String getNombre()*.
   - Método que modifica el nombre de una asignatura: *void setNombre(String nombre)*.
   - Método que devuelve la calificación en la asignatura: *double getCalificacion()*.
   - Método que modifica la calificación: *void setCalificacion(double calificacion)*.
   - Método *void mostrar ()*, que visualiza en pantalla la asignatura con el formato que indica el siguiente ejemplo:

```
Estructuras de datos (6,7)
```

2. Modificar la clase ***NodoPila***.

   Se modificará la clase *NodoPila* proporcionada de forma que el dato del nodo sea un objeto de la clase *Asignatura*. (Es necesario modificar *getDato* y *setDato*).

3. Modificar la clase ***Pila***.

   Se modificará la clase *Pila* de forma que los nodos pasen a contener como dato un objeto de la clase *Asignatura*. Para ello, es necesario modificar los métodos *apilar, desapilar, getCima* y *mostrar* (este último, deberá utilizar el método *mostrar* de la clase *Asignatura*).

4. Los métodos *prepararPila* y *pruebasPila* de la clase ***Principal*** realizan las siguientes tareas:
   - En *prepararPila* se instancian 4 objetos de la clase *Asignatura* con los siguientes datos:

	nombre	calificación
0	Fundamentos Físicos de la Informática	3.50
1	Estructuras de Datos	6.35
2	Aspectos Éticos y Sociales	7.70
3	Fundamentos de seguridad	6.75

- Se instancia un objeto de la clase *Pila* y se añaden las asignaturas anteriores. Dicha pila se devuelve como resultado.
- En el método *pruebasPila*, se utiliza el método *mostrar* de la clase *Pila*, tras lo cual se obtendrá el siguiente resultado:

```
Asignaturas:
      Fundamentos de seguridad (6,8)
      Aspectos Éticos y Sociales (7,7)
      Estructuras de Datos (6,4)
      Fundamentos Físicos de la Informática (3,5)
```

5. Completar los siguientes métodos en la clase *AlgoritmosPila* y después probar su funcionamiento:

   - `public double notaMínima (Pila asignaturas);`

   Dada una pila de asignaturas, devuelve como resultado la calificación mínima en cualquiera de las asignaturas. Si la pila está vacía, devolverá 0 como resultado.

   - `public Asignatura asignaturaNotaMáxima(Pila asignaturas);`

   Dada una pila de asignaturas, devuelve como resultado la asignatura que tiene la calificación máxima en cualquiera de las asignaturas. Si la pila estuviera vacía, se devolverá como resultado *null*.

   - `public void mostrarAprobadas (Pila asignaturas);`

   Dada una pila de asignaturas, muestra por pantalla únicamente las asignaturas que tienen una calificación mayor o igual a 5.

   En los tres métodos anteriores, **es obligatorio manejar la pila de manera recursiva**, y al finalizar la ejecución del método **la pila deberá contener el contenido original sin modificaciones**.

   Ejemplo de funcionamiento para las pruebas que hay que realizar:

```
Asignaturas:
    Fundamentos de seguridad (6,8)
    Aspectos Éticos y Sociales (7,7)
    Estructuras de Datos (6,4)
    Fundamentos Físicos de la Informática (3,5)
Calificación mínima: 3.5
Asignatura con calificación máxima:
Asignatura con nota máxima:
    Aspectos Éticos y Sociales (7,7)
Asignaturas aprobadas:
    Fundamentos de seguridad (6,8)
    Aspectos Éticos y Sociales (7,7)
    Estructuras de Datos (6,4)
```

4. Algoritmos con Colas.

A lo largo de esta parte de la práctica, se va a trabajar con una cola de alumnos. Para ello, primero se preparará a clase *Alumno* y a continuación se modificará la clase Cola.

1. Definir la clase ***Alumno***.

   Cada objeto de la clase *Alumno* contendrá encapsulados los datos de un alumno, incluyendo nombre, número de matrícula, asignaturas en las que está matriculado y calificación media. Para almacenar estos datos, se decide utilizar los siguientes **atributos** en la clase *Alumno*:

   - *nombre y matricula* de tipo *String*.
   - *calificaciónMedia*: de tipo *double.*
   - *numAsignaturas*, de tipo *int*, que contiene el número de asignaturas en las que está matriculado.
   - *asignaturas*, que será una *Pila* de la clase utilizada en el apartado anterior.

   La interfaz de esta clase, es decir, las operaciones que ofrece la clase *Alumno*, vienen definidas por los siguientes **métodos**:

   - Constructor con dos parámetros para inicializar los atributos: *nombre* y *matrícula.* Además se inicializará *calificaciónMedia* con 0 y se preparará la *Pila* de *asignaturas* para su utilización.
   - Método que devuelve el nombre del alumno: *String getNombre()*.
   - Método que modifica el nombre de un alumno: *void setNombre(String nom)*.
   - Método que devuelve la calificación del alumno: *double getCalificacionMedia()*.
   - Método que permite añadir una asignatura al alumno*: void anadirAsignatura(Asignatura asignatura)*. Un alumno puede tener asignaturas repetidas. Por tanto, al añadir una signatura, no hay que comprobar si ya existía. Hay que tener en cuenta que, cada vez que se añada una nueva asignatura al alumno, es necesario recalcular la calificación media.
   - Método que devuelve el número de asignaturas en las que está matriculado el alumno: *int getNumAsignaturas()*.
   - Método *void mostrarAlumno()*, que visualiza en pantalla los datos del alumno, incluyendo las asignaturas en las que está matriculado el alumno con el formato que indica el siguiente ejemplo:

```
Margarita Lopez Medina. Matr: mj7726 (7,5)
Asignaturas:
    Estructuras de Datos (7,5)
```

   Si el alumno no está matriculado en ninguna asignatura, este método visualizará el mensaje: "No está matriculado en ninguna asignatura"

2. Modificar la clase ***NodoCola***.

   Se modificará dicha clase de forma que el dato del nodo sea un objeto de la clase *Alumno*. (Es necesario modificar *getDato* y *setDato*).

3. Modificar la clase ***Cola***.

   Se modificará la clase *Cola* de forma que los nodos contengan como dato un objeto de la clase *Alumno*. Para ello, es necesario modificar los métodos *encolar, desencolar, getPrimero* y *mostrar* (este último, deberá utilizar el método *mostrar* de la clase *Alumno*).

   Ejemplo de funcionamiento del método mostrar de la clase Cola:

```
Contenido del grupo:
Felipe Arias Gonzalez. Matr: aa1253 (6,4)
Asignaturas:
    Estructuras de Datos (6,4)
Manuel Garcia Sacedón. Matr: ax0074 (4,4)
Asignaturas:
    Estructuras de Datos (4,4)
Margarita Lopez Medina. Matr: mj7726 (7,5)
Asignaturas:
    Estructuras de Datos (7,5)
Estela Sanchez Arellano. Matr: bc2658 (5,9)
Asignaturas:
    Fundamentos de seguridad (6,8)
    Álgebra (5,0)
```

4. El método *prepararCola* de la clase ***Principal*** realiza las siguientes tareas:
   - Instancia 4 objetos de la clase *Alumno* con los siguientes datos:

	nombre	matrícula
0	Felipe Arias González	aa1253
1	Manuel García Sacedón	ax0074
2	Margarita López Medina	mj7726
3	Estela Sánchez Arellano	bc2658

   - Se matricula a los cuatro alumnos en “Estructuras de Datos”. Además, a “Estela Sánchez Arellano” en “Algebra” y “Fundamentos de Seguridad”.
   - Se instancia una ***Cola*** de alumnos y se insertan los alumnos anteriores.

5. Completar los siguientes métodos en la clase ***AlgoritmosCola***, y verificar su correcto funcionamiento:

   - **public void** mostrarGrupo (Cola grupo)

     Muestra en pantalla el número de alumnos del grupo así como los nombres de todos los alumnos que lo componen con un número de orden, según el formato que se indica en el siguiente ejemplo:

     ```
     El grupo contiene 4 alumnos
     1. Felipe Arias Gonzalez
     2. Manuel Garcia Sacedón
     3. Margarita Lopez Medina
     4. Estela Sanchez Arellano
     ```

   - **public** Cola alumnosAprobados (Cola grupo)

     Crea una nueva ***Cola*** con los alumnos que tienen de media una calificación igual o superior a 5 y la devuelve como resultado.

   - **public double** calificaciónMedia (Cola grupo)

En los tres métodos anteriores, **es obligatorio manejar la cola de manera iterativa**, y al finalizar la ejecución del método **la cola deberá contener el contenido original sin modificaciones**.

6. Ejemplo de funcionamiento con las pruebas solicitadas:

```
El grupo contiene 4 alumnos
1. Felipe Arias Gonzalez
2. Manuel Garcia Sacedón
3. Margarita Lopez Medina
4. Estela Sanchez Arellano
Calificación media del grupo: 6.01875

Alumnos aprobados:
Contenido del grupo:
Felipe Arias Gonzalez. Matr: aa1253 (6,4)
Asignaturas:
    Estructuras de Datos (6,4)
Margarita Lopez Medina. Matr: mj7726 (7,5)
Asignaturas:
    Estructuras de Datos (7,5)
Estela Sanchez Arellano. Matr: bc2658 (5,9)
Asignaturas:
    Fundamentos de seguridad (6,8)
    Álgebra (5,0)

Calificación media de los aprobados: 6.575
```