Cohesión

La Cohesión de la clase **Alumno** es adecuada, tiene un nombre apropiado y su único atributo pertenece a un **Alumno**.

Alumno.java

```java
package uam.patrones.grasp.clases;

public class Alumno {

	private String matricula;

	public String getMatricula() {
		return matricula;
	}

	public void setMatricula(String matricula) {
		this.matricula = matricula;
	}

	public String toString() {
		return matricula;
	}
}
```

Lo mismo sucede con la clase **Uea** que tiene un nombre apropiado y su atributo representa a una **Uea**.

Uea.java

```java
package uam.patrones.grasp.clases;

public class Uea {

	private String clave;

	public String getClave() {
		return clave;
	}

	public void setClave(String clave) {
		this.clave = clave;
	}

	public String toString() {
		return clave;
	}
}
```

La clase **QueEstaCursando** no tiene buena Cohesión, ya que los nombres de la clase y de al menos un método no son tan adecuados, adicionalmente, no cumple con el encapsulamiento, ya que sus atributos son públicos. Se propone:

- Cambiar el nombre de la clase
- Cambiar el nombre de un atributo
- Cambiar la visibilidad a privada
- Generar métodos *get()* y *set()*

AlumnoUea.java

```
package uam.patrones.grasp.clases;

import java.util.LinkedList;

public class AlumnoUea {

	private Alumno alumno;
	private LinkedList<Uea>listaUeas;

	public Alumno getAlumno() {
		return alumno;
	}
	public void setAlumno(Alumno alumno) {
		this.alumno = alumno;
	}
	public LinkedList<Uea> getListaUeas() {
		return listaUeas;
	}
	public void setListaUeas(LinkedList<Uea> listaUeas) {
		this.listaUeas = listaUeas;
	}
}
```

La clase **OperacionesBaseDeDatos** tiene un método (*imprimirDatosConsultados*) que imprime los resultados, por lo que está realizando diversas tareas no relacionadas entre sí, para aumentar su cohesión, será necesario moverlo a la clase encargada de las operaciones de Alumno y Uea.

ServiciosAlumnoUea.java

```
public void imprimirDatosConsultados(Alumno alumno, LinkedList<Uea>lista)
{
		System.out.println(alumno.toString());
		System.out.println(lista);
}
```

También se observa que hay una consulta de Alumno y una de Uea.

Después de los cambios, en esta clase, si bien podría considerarse que la conexión puede abrirse ahí, también sería adecuado cambiar el método a una nueva clase más específica. El resto de los métodos si se relacionan con obtener la información de la base de datos de un **Alumno** y una **Uea**.

Correcciones sugeridas:

- Crear una clase solo para la conexión
- Dejar solo métodos que permitan obtener la información de Alumno, Uea y Lista de Uea

AdministrarConexion.java

```
package uam.patrones.grasp.baseDatos;

public class AdministrarConexion {

	public void abrirConexion() {
		System.out.println("Se abrió la conexión");
	}
}
```

El crear una clase para la conexión, permite que las clase encargada de solicitar el alumno, en particular **ServiciosAlumnoUea** ya no deba preocuparse por cómo abrir la conexión, llamando solo a la clase encargada del manejo de base de datos.

OperacionesBaseDatos.java

```
package uam.patrones.grasp.operaciones;

import java.util.LinkedList;

import uam.patrones.grasp.baseDatos.AdministrarConexion;
import uam.patrones.grasp.clases.Alumno;
import uam.patrones.grasp.clases.Uea;

public class OperacionesBaseDatos {


	public Alumno consultarAlumno(String matricula) {
		AdministrarConexion conexion = new AdministrarConexion();
		conexion.abrirConexion();
		Alumno alumno = new Alumno();
		alumno.setMatricula(matricula);
		return alumno;
	}

	public Uea consultarUea(String clave) {
		AdministrarConexion conexion = new AdministrarConexion();
		conexion.abrirConexion();
		Uea uea = new Uea();
		uea.setClave(clave);
		return uea;
	}

	public LinkedList<Uea>listaUeas(Alumno alumno) {

		AdministrarConexion conexion = new AdministrarConexion();
		conexion.abrirConexion();
```

```
            LinkedList<Uea>listaUea = new LinkedList<Uea>();
            Uea uea1 = new Uea();
            uea1.setClave("1234");

            Uea uea2 = new Uea();
            uea2.setClave("2345");

            listaUea.add(uea1);
            listaUea.add(uea2);

            return listaUea;
      }
}
```

Quedará pendiente analizar si es correcto que cada método creé un objeto para la conexión.

Creador

- La clase **AlumnoUea** debería crear objetos de tipo **Alumno** y **Uea**
- La clase **ServiciosAlumnoUea** necesita crear objetos de tipo **AlumnoUea** y de tipo **OperacionesBaseDatos**.
- La clase **OperacionesBaseDatos** debería crear objetos de tipo **AdministrarConexion**
- La clase **Principal** solo debería crear objetos de la clase **ServiciosAlumno**

Sin embargo, al invocar al método *consultarTiraMaterias()*, se le debe enviar un objeto tipo Alumno, ¿es necesario crearlo?, ¿es necesario si quiera enviarlo?

Acoplamiento

Las clases que tienen más problemas con el acoplamiento son las de **Principal** y **ServiciosAlumnoUea**, ya que llaman a otras clases para realizar sus tareas.

La clase Principal solo debería llamar al método de **ServiciosAlumnoUea** que haga la consultar

Principal.java

```
package uam.patrones.grasp.principal;

import uam.patrones.grasp.operaciones.ServiciosAlumnoUea;

public class Principal {

	public static void main(String[] args) {

		ServiciosAlumnoUea servicios = new ServiciosAlumnoUea();
		servicios.consultarTiraMaterias("matricula");

	}
}
```

Acoplamiento entre Métodos

Si se revisan los métodos de las clases, también se encuentra que hay mucha comunicación entre ellos que no es necesaria.

Considerar la clase **ServiciosAlumnoUea**:

- El método *consultarDatosAlumno()* es correcto, ya que solo se encarga de recuperar a un alumno (independientemente de lo que reciba)
- El método *consultarTiraMaterias()* manda a llamar al método consultar **Alumno** a pesar de que recibe uno, esto no es adecuado. Si bien necesita la información de un Alumno, puede obtenerla de la base de datos.
- El método *consultarUea()* es correcto, ya que solo se encarga de recuperar a una uea (independientemente de lo que reciba)
- El método *imprimirDatosConsultados()* es correcto, ya que su única función es imprimir los resultados (independientemente de lo que reciba)

ServiciosAlumnoUea.java

```
package uam.patrones.grasp.operaciones;

import uam.patrones.grasp.clases.AlumnoUea;

public class ServiciosAlumnoUea {

	private OperacionesBaseDatos base;
	private AlumnoUea alumnoUea;
```

```
	public ServiciosAlumnoUea() {
		base = new OperacionesBaseDatos();
		alumnoUea = new AlumnoUea();
	}

	public void consultarTiraMaterias(String matricula){
		alumnoUea.setAlumno(base.consultarAlumno(matricula));

	alumnoUea.setListaUeas(base.listaUeas(alumnoUea.getAlumno()));
		imprimirDatosConsultados();
	}

	private void imprimirDatosConsultados() {
		System.out.println(alumnoUea.getAlumno().toString());
		System.out.println(alumnoUea.getListaUeas());
	}

	public AlumnoUea getAlumnoUea() {
		return alumnoUea;
	}


}
```

El que no todos los atributos o métodos sean públicos, tiene que ver con si otras clases realmente necesitan acceso a ellos, incluso si necesitan acceso a los métodos *get()/set()*.

Finalmente se separó la funcionalidad de consultar alumnos y ueas.

ServiciosAlumno.java

```
package uam.patrones.grasp.operaciones;

import uam.patrones.grasp.clases.Alumno;

public class ServiciosAlumno {

	private Alumno alumno;
	private OperacionesBaseDatos base;


	public ServiciosAlumno() {
		alumno = new Alumno();
		base = new OperacionesBaseDatos();
	}

	public void consultarDatosAlumno(String matricula) {
		alumno = base.consultarAlumno("matricula");
```

```
	}

	public Alumno getAlumno() {
		return alumno;
	}

}
```

ServiciosUea.java

```
package uam.patrones.grasp.operaciones;

import uam.patrones.grasp.clases.Uea;

public class ServiciosUea {

	private Uea uea;
	private OperacionesBaseDatos base;

	public ServiciosUea() {
		uea = new Uea();
		base = new OperacionesBaseDatos();
	}

	public Uea consultarUea(String clave) {
		uea = base.consultarUea(clave);
		return uea;
	}

	public Uea getUea() {
		return uea;
	}


}
```

¿Y si en la clase Principal se quisiera recuperar los datos para enviarlos a otra clase? Estos datos ya forman parte de las clases correspondientes.

Principal.java

```
package uam.patrones.grasp.principal;

import uam.patrones.grasp.operaciones.ServiciosAlumnoUea;

public class Principal {

	public static void main(String[] args) {

		ServiciosAlumnoUea servicios = new ServiciosAlumnoUea();
```

```
            servicios.consultarTiraMaterias("matricula");

      System.out.println(servicios.getAlumnoUea().getAlumno().toString());
      System.out.println(servicios.getAlumnoUea().getListaUeas());

      }
}
```