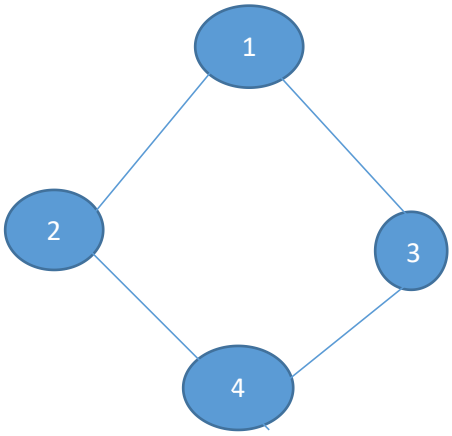
```
//Clase Sistema
public void agregarAlumno(Alumno nuevo)
        throws ClaveYaExistenteException, DatoInvalidoException
        {
        if (!Persona.validaPersona(nuevo))  1
                throw new DatoInvalidoException(nuevo, "Se encontraron datos inválidos."); 2
        else
        {
                nuevo.setLegajo(Alumno.getNuevoLegajo()); 3
                this.alumnos.agregar(nuevo);
        }
        } 4
```



Complejidad ciclomatica

Complejidad ciclomatica= 4-4+2=2

C1 : 1-2-4

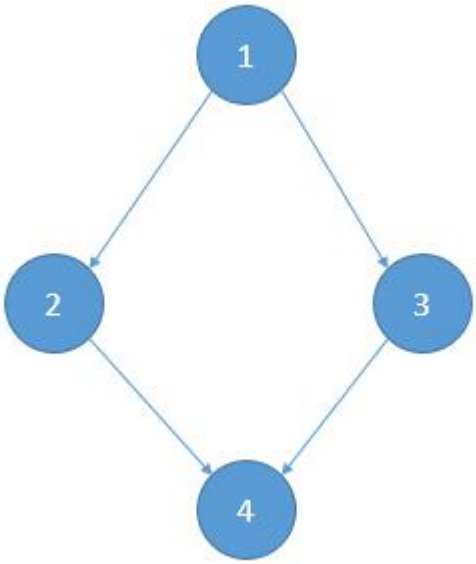
C2:1-3-4

Id	Objetivo de	Datos de	Procedimiento	Salida esperada	Resultado

	la prueba	entrada			
T1	Verificar el camino 1	Objeto Alumno	1- Crear Alumno A(“Juan Pico”,”Falucho 34333”,”aaaaa”) 2-Ejecutar prueba	DatoInvalidoException	PASS
T2	Verificar el camino 2	Objeto Alumno	1- Crear Alumno A(“Juan Pico”,”Falucho 34333”,juanferpjco@gmail.com) 2-Ejecutar prueba	Alumno agregado correctamente	PASS

//Clase Sistema

```
public void agregarProfesor(Profesor nuevo)
    throws ClaveYaExistenteException, DatoInvalidoException
  {
    if (!Persona.validaPersona(nuevo)) 1
        throw new DatoInvalidoException(nuevo, "Se encontraron datos inválidos."); 2
    else
    {
        nuevo.setLegajo(Profesor.getNuevoLegajo());
        this.profesores.agregar(nuevo);  3
    }
  } 4
```



Complejidad ciclomatica

Complejidad ciclomatica= 4-4+2=2

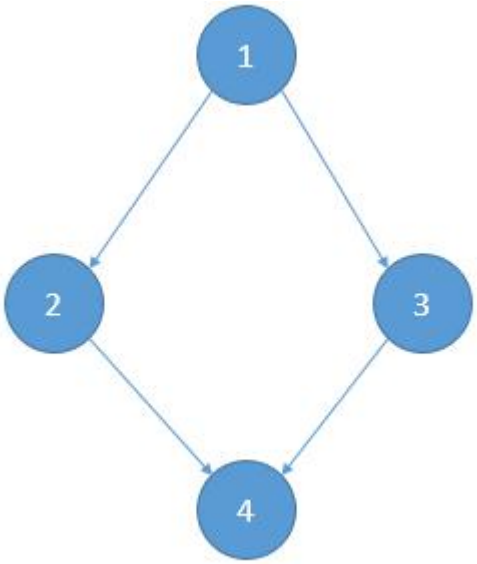
C1 : 1-2-4

C2:1-3-4

Id	Objetivo de la prueba	Datos de entrada	Procedimiento	Salida esperada	Resultado
T1	Verificar el camino 1	Objeto Profesor	1-Crear Profesor A(“Juan Pico”,”Falucho 34333”,”aaaaa”,”22352522”) 2-Ejecutar prueba	DatoInvalidoException	PASS
T2	Verificar el camino 2	Objeto Profesor	1-Crear Profesor A(“Juan Pico”,”Falucho 34333”,juanferpjco@gmail.com,” 2235257381“) 2-Ejecutar prueba	Profesor agregado correctamente	PASS

//Clase Sistema

```
public void agregarAsignatura(Asignatura nuevo)
    throws ClaveYaExistenteException, DatoInvalidoException
  {
    if (!Asignatura.validaAsignatura(nuevo)) 1
        throw new DatoInvalidoException(nuevo, "Se encontraron datos inválidos."); 2
    else
    {
        nuevo.setIdentificacion(Asignatura.getNuevaIdentificacion());
        this.planDeEstudio.agregar(nuevo); 3
    }
  } 4
```



Complejidad ciclomatica

Complejidad ciclomatica= 4-4+2=2

C1 : 1-2-4

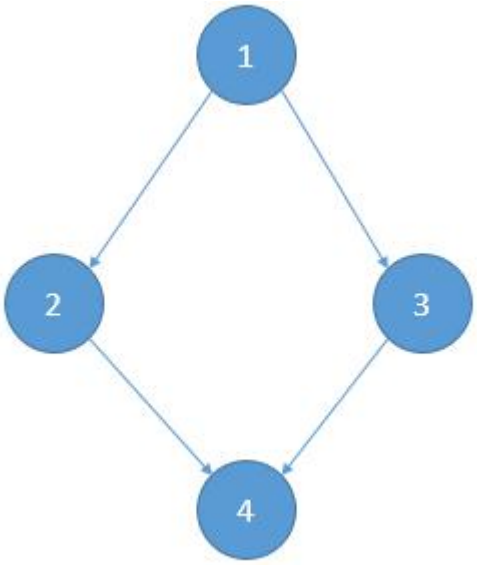
C2:1-3-4

Id	Objetivo de la prueba	Datos de entrada	Procedimiento	Salida esperada	Resultado
T1	Verificar el camino 1	Objeto Asignatura	1-Crear Asignatura A("") 2-Ejecutar prueba	DatoInvalidoException	PASS
T2	Verificar el camino 2	Objeto Asignatura	1-Crear Asignautra A("mateA") 2-Ejecutar prueba	Asignatura agregada correctamente	PASS

//Clase Sistema

```
public void agregarCursada(Cursada nuevo)
    throws ClaveYaExistenteException, DatoInvalidoException
  {
    if (!Cursada.validaCursada(nuevo)) 1
        throw new DatoInvalidoException(nuevo, "Se encontraron datos inválidos."); 2
    else
    {
```

```
            nuevo.setIdentificacion(Cursada.getNuevaIdentificacion());
            this.calendario.agregar(nuevo); 3
        }
    } 4
```



Complejidad ciclomatica

Complejidad ciclomatica= 4-4+2=2

C1 : 1-2-4

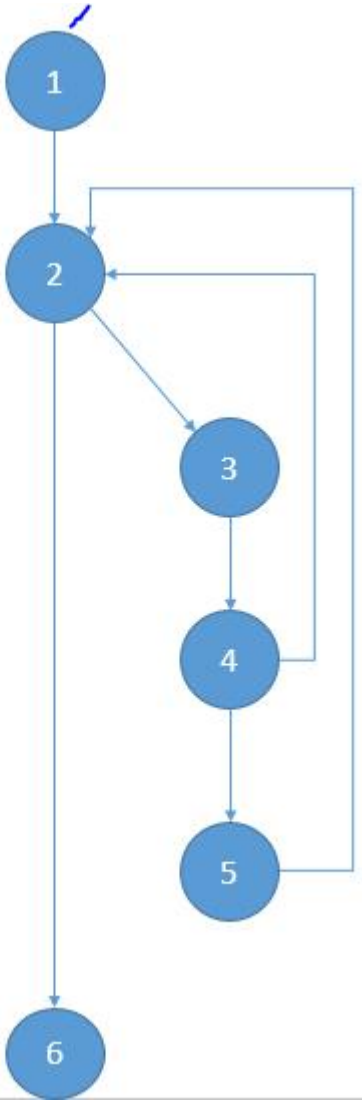
C2:1-3-4

Id	Objetivo de la prueba	Datos de entrada	Procedimiento	Salida esperada	Resultado
T1	Verificar el camino 1	Objeto Cursada	1-Crear Cursada A(mateA,"ccccc",LUNES,"8","10") 2-Ejecutar prueba	DatoInvalidoException	PASS
T2	Verificar el camino 2	Objeto Cursada	1-Crear Cursada A(mateA,"01-2017",LUNES,"8","10")) 2-Ejecutar prueba	Cursada agregada correctamente	PASS

//Clase Sistema

```
public void eliminarAlumno(Alumno elim)
    {
```

```
    Cursada aux;
    Iterator<Cursada> it;
    this.alumnos.eliminar(elim);
    it = this.calendario.elementosPorClavePrimaria(); 1
    while (it.hasNext()) 2
    {
        aux = it.next(); 3
       if (aux.tieneAlumno(elim)) 4
          aux.bajaAlumno(elim); 5
    }
} 6
```



Complejidad Ciclomatica

Complejidad Ciclomatica= 7-6+2=3

C1 : 1-2-6

C2: 1-2-3-4-2-6

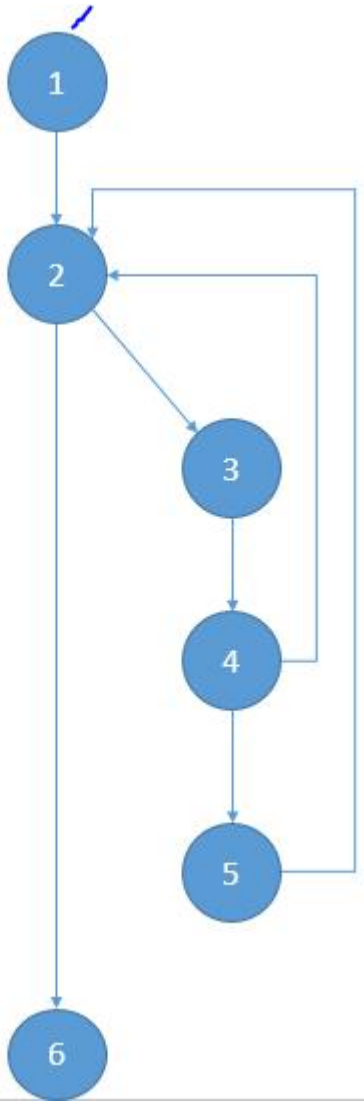
C3:1-2-3-4-5-2-6

Id	Objetivo de la prueba	Datos de entrada	Procedimiento	Salida esperada	Resultado
T1	Verificar el camino 1	Objeto Alumno	1-Crear Alumno A("Juan Pico","Falucho 34333","aaaaa@gmail.com") 2-Ejecutar prueba	Alumno eliminado(en este caso no había cursadas en el sistema)	PASS
T2	Verificar el camino 2	Objeto Alumno	1-Crear Alumno A("Juan Pico","Falucho 34333","aaaaa@gmail.com") 2-Crear cursada y agregarla al sistema (una o muchas) 3-Ejecutar prueba	Alumno eliminado(en este caso había cursadas en el sistema, pero el alumno no estaba en ninguna)	PASS
T3	Verificar el camino 3	Objeto Alumno	1-Crear Alumno A("Juan Pico","Falucho 34333","aaaaa@gmail.com") 2-Crear cursada y agregarla al sistema (una o muchas) 3-Agregar el alumno a una/muchas cursadas 4-Ejecutar prueba	Alumno eliminado(en este caso había cursadas en el sistema y el alumno estaba en una o en muchas)	PASS

//Clase Sistema

```
public void eliminarProfesor(Profesor elim)
  {
     Cursada aux;
     Iterator<Cursada> it;
     this.profesores.eliminar(elim);
     it = this.calendario.elementosPorClavePrimaria(); 1
     while (it.hasNext()) 2
     {
```

```
                aux = it.next();3
        if (aux.tieneProfesor(elim) ) 4
                aux.bajaProfesor(elim); 5
    }
} 6
```



Complejidad Ciclomatica

Complejidad Ciclomatica= 7-6+2=3

C1 : 1-2-6

C2: 1-2-3-4-2-6

C3:1-2-3-4-5-2-6

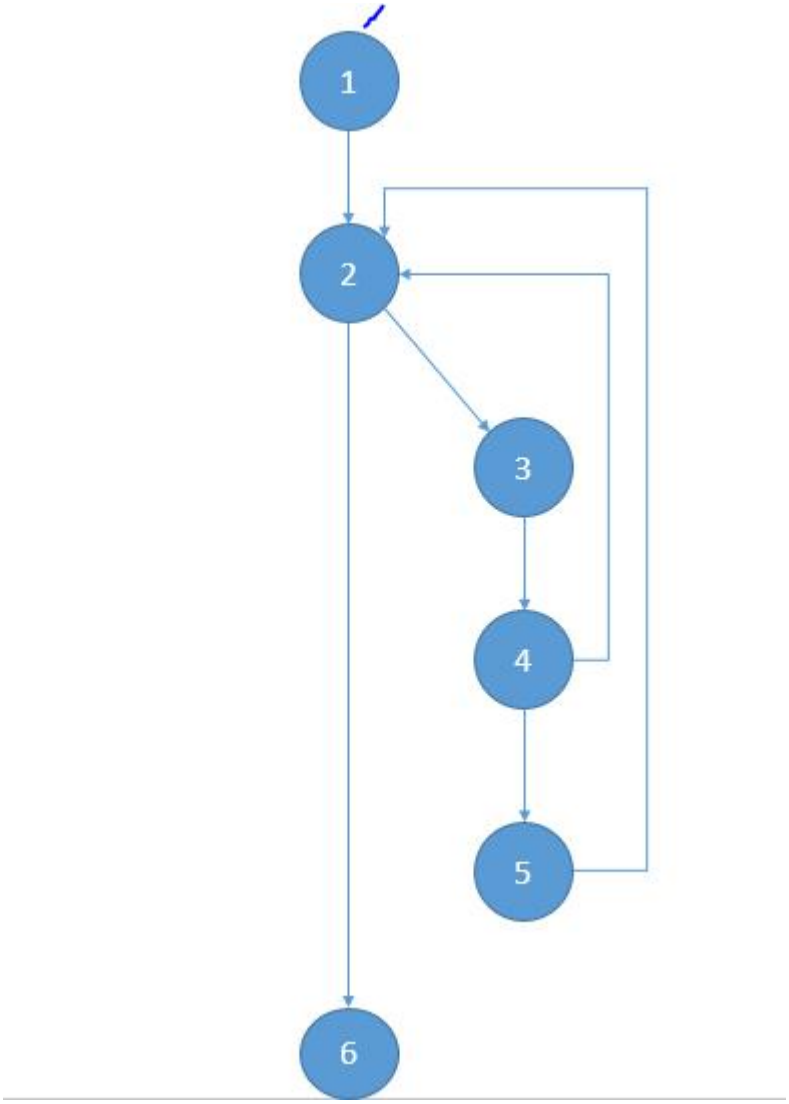
Id	Objetivo de la	Datos de entrada	Procedimiento	Salida esperada	Resultado

	prueba				
T1	Verificar el camino 1	Objeto Profesor	1-Crear Profesor A("Juan Pico","Falucho 34333",aaaaa@gmail.com,"2235257381") 2-Ejecutar prueba	Profesor eliminado(en este caso no había cursadas en el sistema)	PASS
T2	Verificar el camino 2	Objeto Profesor	1-Crear Profesor A("Juan Pico","Falucho 34333",aaaaa@gmail.com,"2235257381") 2-Crear cursada y agregarla al sistema (una o muchas) 3-Ejecutar prueba	Profesor eliminado(en este caso había cursadas en el sistema, pero el Profesor no estaba en ninguna)	PASS
T3	Verificar el camino 3	Objeto Profesor	1-Crear Profesor A("Juan Pico","Falucho 34333",aaaaa@gmail.com,"2235257481") 2-Crear cursada y agregarla al sistema (una o muchas) 3-Agregar el Profesor a una/muchas cursadas 4-Ejecutar prueba	Profesor eliminado(en este caso había cursadas en el sistema y el Profesor estaba en una o en muchas)	PASS

//Clase Sistema

```
public void eliminarAsignatura(Asignatura elim)
  {
     Cursada aux;
     Iterator<Cursada> it;
     this.planDeEstudio.eliminar(elim);
     this.eliminaAsignaturaEnAlumnos(elim);
     this.eliminaAsignaturaEnProfesores(elim);
     this.eliminaAsignaturaEnCorrelatividades(elim);
     it = this.calendario.elementosPorClavePrimaria(); 1
     while (it.hasNext()) 2
     {
                aux = it.next(); 3
```

```
        if (aux.getAsignatura().equals(elim)) 4
            this.eliminarCursada(aux); 5
    }
} 6
```



Complejidad Ciclomatica

Complejidad Ciclomatica= 7-6+2=3

C1 : 1-2-6

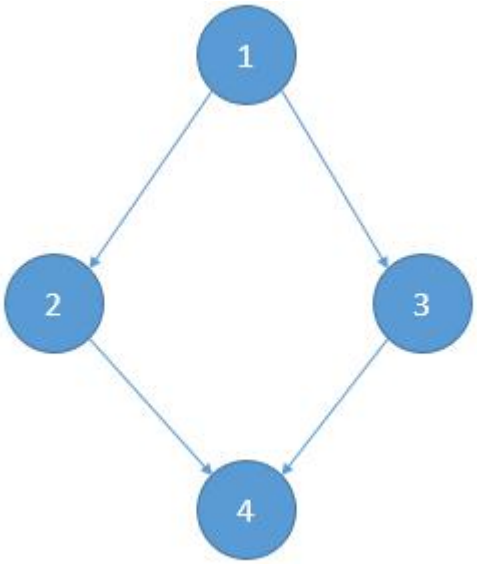
C2: 1-2-3-4-2-6

C3:1-2-3-4-5-2-6

Id	Objetivo de la prueba	Datos de entrada	Procedimiento	Salida esperada	Resultado
T1	Verificar el	Objeto	1-Crear Asignatura	Asignatura	PASS

	camino 1	Asignatura	A(“mateA”) 2-Ejecutar prueba	eliminado(en este caso la asignatura no correspondia a ninguna cursada)	
T2	Verificar el camino 2	Objeto Asignatura	1-Crear Asignatura A(“mateA”) 2-Crear cursada y agregarla al sistema (una o muchas), sin tenes a A como asignatura 3-Ejecutar prueba	Asignatura eliminada(en este caso había cursadas en el sistema, pero ninguna correspondia a la asignatura)	PASS
T3	Verificar el camino 3	Objeto Asignatura	1-Crear Asignatura A(“mateA”) 2-Crear cursada y agregarla al sistema (una o muchas), estas tienen que tenes como asignatura a A 4-Ejecutar prueba	Asignatura eliminadoa(en este caso había cursadas en el sistema y una o muchas correspondían a la asignatura)	PASS

```
//Clase Asignatura
public void agregarCorrxelativa(Asignatura correlativa)
    throws ClaveYaExistenteException, DatoInvalidoException
  {
    if (this.equals(correlativa)) 1
        throw new DatoInvalidoException(correlativa, "La asignatura no puede ser correlativa de si misma."); 2
    else
        this.correlatividades.agregar(correlativa); 3
  } 4
```



Complejidad ciclomatica

Complejidad ciclomatica= 4-4+2=2

C1 : 1-2-4

C2:1-3-4

Id	Objetivo de la prueba	Datos de entrada	Procedimiento	Salida esperada	Resultado
T1	Verificar el camino 1	Objeto Asignatura	1-Crear Asignatura A(“mateA”) 2-Ejecutar prueba, pasando como parámetro mateA	DatoInvalidoException	PASS
T2	Verificar el camino 2	Objeto Asignatura	1-Crear Asignatura A(“mateA”) 2-Crear Asignatura B(“mateB”) 3-Ejecutar prueba pasando como parámetro mateB	Correlativa agregada correctamente	PASS

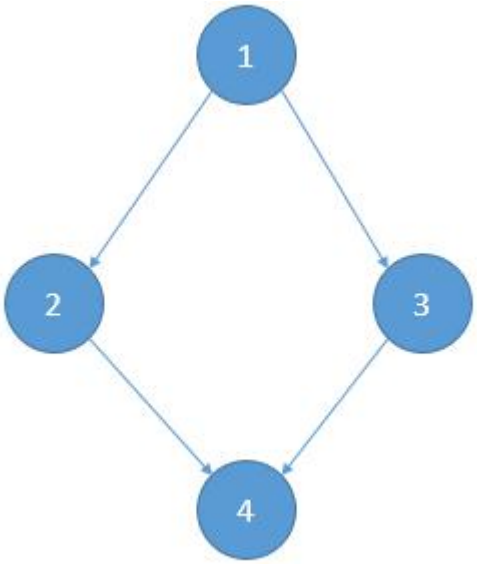
(Metodo de la clase IndicePrimario que se usa en varios casos)

```
public void agregar(V nuevo)
    throws ClaveYaExistenteException
  {
    if (this.contieneClave(nuevo.getClavePrimaria())) 1
      throw new ClaveYaExistenteException(nuevo.getClavePrimaria()); 2
```

else

this.elementos.put(nuevo.getClavePrimaria(), nuevo); 3

} 4



Complejidad ciclomatica

Complejidad ciclomatica= 4-4+2=2

C1 : 1-2-4

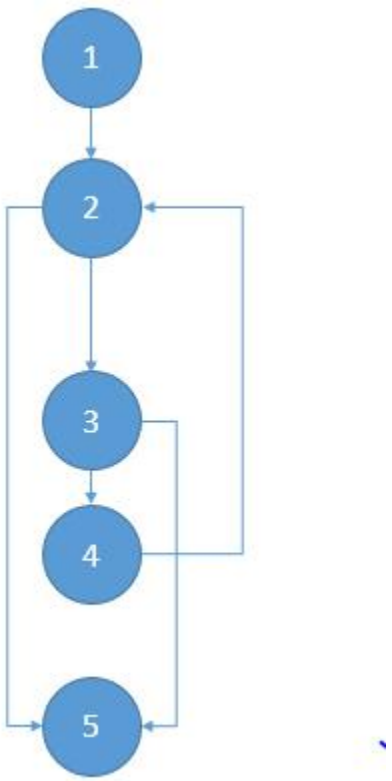
C2:1-3-4

//En la prueba de este método se toma uno de los métodos que lo utilizan en este caso agregarCompetencia para probarlo

Id	Objetivo de la prueba	Datos de entrada	Procedimiento	Salida esperada	Resultado
T1	Verificar el camino 1	Objeto V	1-Crear Asignatura A(“mateA”) 2-Crear Profesor P 3-Llamar al método agregarCompetencia con mateA 4-Volver a llamar al método agregarCompetencia con mateA	ClaveYaExistenteException	PASS
T2	Verificar el camino 2	Objeto V	1-Crear Asignatura A(“mateA”) 2-Crear Profesor P 3-Llamar al método	Competencia agregada correctamente	PASS

			agregarCompetencia con mateA		

```
//Clase sistema
public boolean alumnoDisponible(Alumno alumno, Cursada cursada)
  {
     boolean res = true;
     Cursada aux;
     Iterator<Cursada> it = this.calendario.elementosPorClavePrimaria(); 1
     while (it.hasNext() 2 && res 3)
     {
         aux = it.next();
        // El alumno no está en la cursada o la misma no debe colisionar con la solicitada
         res = !aux.tieneAlumno(alumno) || !aux.hayColision(cursada); 4
     }
     return res;
  } 5
```



Complejidad Ciclomatica

Complejidad Ciclomatica= 6-5+2=3

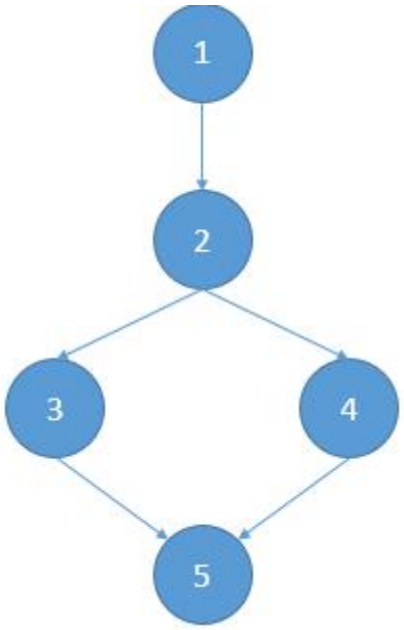
C1: 1-2-5

C2:1-2-3-5

C3: 1-2-3-4-2-5

Id	Objetivo de la prueba	Datos de entrada	Procedimiento	Salida esperada	Resultado
T1	Verificar el camino 1	Objeto Alumno, Objeto Cursada	1-Crear Alumno A valido 2-Crear Cursada C valida 3-ejecutar prueba	True (La cursada no estaba agreada al sistema)	PASS
T2	Verificar el camino 2	Objeto Alumno, Objeto Cursada	1-Crear Alumno A valido 2-Crear Cursada C valida 3-Agrego A a C 4-Ejecuto prueba	False(el alumno ya esta en la cursada, no esta disponible)	PASS
T3	Verificar el camino 3	Objeto Alumno, Objeto Cursada	1-Crear Alumno A valido 2-Crear Cursada C valida 3-Agrego la cursada y el alumno al sistema 4-Ejecuto prueba	True(el alumno esta disponible)	PASS

```
//Clase Sistema
public Iterator<Alumno> buscarAlumno(String nombre)
    throws NoEncontradoException
  {
    Iterator<Alumno> ret = this.busquedaPorNombre(nombre, this.alumnos); 1
    if (!ret.hasNext()) 2
        throw new NoEncontradoException(nombre, "El nombre solicitado no fue encontrado."); 3
    else
        return ret; 4
  }  5
```



Complejidad Ciclomatica

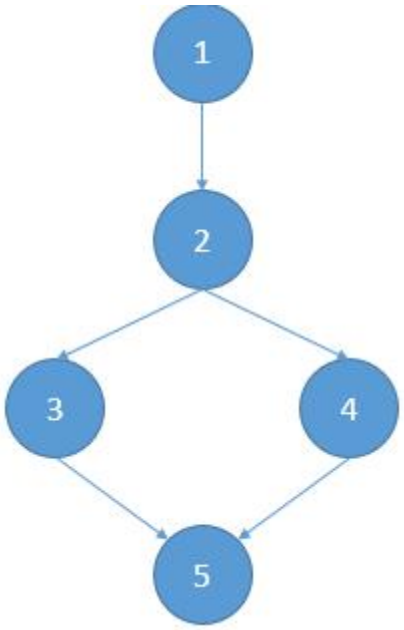
Complejidad Ciclomatica= 5-5+2=2

C1: 1-2-3-5

C2: 1-2-4-5

Id	Objetivo de la prueba	Datos de entrada	Procedimiento	Salida esperada	Resultado
T1	Verificar el camino 1	Objeto String	1-Crear Alumno A valido 2-Ejecutar prueba	NoEncontradoException	PASS
T2	Verificar el camino 2	Objeto String	1-Crear Alumno A valido 2-Añadir alumno al sistema 3-Ejecutar pruba	Iterator<Alumno> (un iterator con todos los alumnos con el mismo String de nombre pasado como entrada)	PASS

```
//Clases sistema
public Iterator<Profesor> buscarProfesor(String nombre)
    throws NoEncontradoException
  {
    Iterator<Profesor> ret = this.busquedaPorNombre(nombre, this.profesores); 1
    if (!ret.hasNext()) 2
      throw new NoEncontradoException(nombre, "El nombre solicitado no fue encontrado."); 3
    else
      return ret; 4
  } 5
```



Complejidad Ciclomatica

Complejidad Ciclomatica= 5-5+2=2

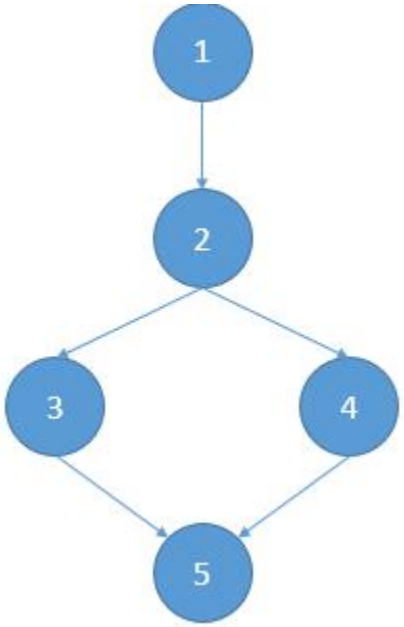
C1: 1-2-3-5

C2: 1-2-4-5

Id	Objetivo de la prueba	Datos de entrada	Procedimiento	Salida esperada	Resultado
T1	Verificar el camino 1	Objeto String	1-Crear Profesor A valido 2-Ejecutar prueba	NoEncontradoException	PASS
T2	Verificar el camino 2	Objeto String	1-Crear Profesor A valido 2-Añadir Profesor al sistema 3-Ejecutar prueba	Iterator< Profesor > (un iterator con todos los Profesores con el mismo String de nombre pasado como entrada)	PASS

```
//Clase sistema
public Iterator<Asignatura> buscarAsignatura(String nombre)
    throws NoEncontradoException
  {
    Iterator<Asignatura> ret = this.busquedaPorNombre(nombre, this.planDeEstudio); 1
    if (!ret.hasNext()) 2
      throw new NoEncontradoException(nombre, "El nombre solicitado no fue encontrado."); 3
    else
      return ret; 4
```

} 5



Complejidad Ciclomatica

Complejidad Ciclomatica= 5-5+2=2

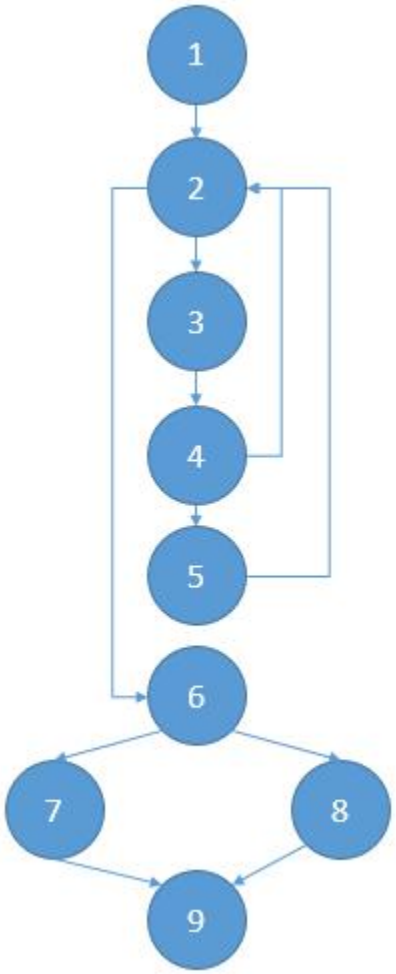
C1: 1-2-3-5

C2: 1-2-4-5

Id	Objetivo de la prueba	Datos de entrada	Procedimiento	Salida esperada	Resultado
T1	Verificar el camino 1	Objeto String	1-Crear Asignatura A valido 2-Ejecutar prueba	NoEncontradoException	PASS
T2	Verificar el camino 2	Objeto String	1-Crear Asignatura A valido 2-Añadir Asignatura al sistema 3-Ejecutar prueba	Iterator< Asignatura > (un iterator con todos las Asignaturas con el mismo String de nombre pasado como entrada)	PASS

```
//Clase sistema
public Iterator<Cursada> buscarCursada(String nombreAsignatura)
    throws NoEncontradoException
  {
    Cursada cursada;
    Iterator<Cursada> cursadas = this.calendario.elementosPorClavePrimaria();
    ArrayList<Cursada> aux = new ArrayList<Cursada>();
    String nombreUpper = nombreAsignatura.toUpperCase(); 1
```

```
    while (cursadas.hasNext()) 2
    {
       cursada = cursadas.next(); 3
       if (cursada.getAsignatura() .getNombre().toUpperCase().contains(nombreUpper)) 4
          aux.add(cursada); 5
    }
    if (aux.isEmpty()) 6
       throw new NoEncontradoException(nombreAsignatura,
                                  "No se encontró ninguna cursada cuya asignatura coincida con el nombre dado."); 7
    else
       return aux.iterator(); 8
} 9
```



Complejidad Ciclomatica

Complejidad Ciclomatica= 11-9 +2=4

C1: 1-2-6-7-9

C2: 1-2-6-8-9

C3: 1-2-3-4-2-6-8-9

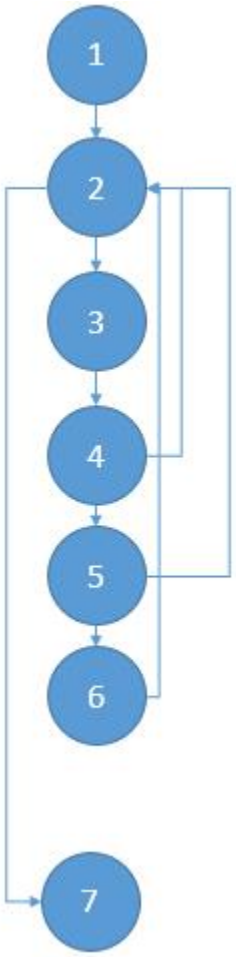
C4: 1-2-3-4-5-2-6-8-9

Id	Objetivo de la prueba	Datos de entrada	Procedimiento	Salida esperada	Resultado
T1	Verificar el camino 1	Objeto String	1-Crear Asignatura A valido 2-Crear Cursada de Asignatura A y no agregarla al sistema 3-Ejecutar prueba	NoEncontradoException	PASS
T2	Verificar el camino 2	Objeto String	Impracticable(si no hay cursadas en el sistema aux va a estar vacio)		
T3	Verificar el camino 3	Objeto String	1-Crear Asignatura A valido 2-Crear Cursada de Asignatura A y agregarla al sistema 3-Ejecutar prueba pasando como parámetro un String que no corresponda al nombre de A	NoEncontradoException	PASS
T4	Verificar el camino 4	Objeto String	1-Crear Asignatura A valido 2-Crear Cursada de Asignatura A y agregarla al sistema 3-Ejecutar prueba pasando como parámetro el nombre de la asigntura A	Iterator<Cursada> (un interator con la cursada con el nombre de la asigntura pasada como parametro)	PASS

//Clase Sistema

```
public void quitarCompetenciaAProfesor(Profesor profesor, Asignatura asignatura)
  {
    Cursada aux;
    Iterator<Cursada> it = this.calendario.elementosPorClavePrimaria(); 1
    while (it.hasNext()) 2
    {
      aux = it.next(); 3
      if (aux.getAsignatura().equals(asignatura) 4 && aux.tieneProfesor(profesor)) 5
        aux.bajaProfesor(profesor); 6
    }
    profesor.eliminarCompetencia(asignatura);
```

} 7



Complejidad Ciclomatica

Complejidad Ciclomatica= 9-7+2=4

C1:1-2-3-4-2-7

C2:1-2-3-4-5-2-7

C3:1-2-3-4-5-6-7

C4:1-2-7

Id	Objetivo de la prueba	Datos de entrada	Procedimiento	Salida esperada	Resultado
T1	Verificar el camino 1	Objeto Profesor,Objeto Asignatura	1-Crear Profesor P valido 2-Crear Asigntura A valida 3-Agregar A como competencia de P 4-Ejecutar prueba , pasando como parámetro una asignatura que no tenga cursadas.	Se elimina la asignatura como competencia del profesor	PASS
T2	Verificar el camino 2	Objeto Profesor,Objeto Asignatura	1-Crear Profesor P valido 2-Crear Asigntura A valida 3-Creo Cursadas de la asignatura A	El profesor no tiene como competencia a esa asignatura	PASS

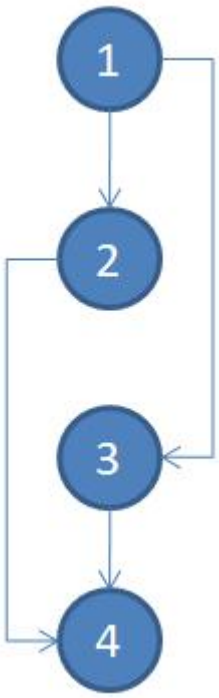
			4-Ejecutar prueba , pasando como parámetro P y A		
T3	Verificar el camino 3	Objeto Profesor,Objeto Asignatura	1-Crear Profesor P valido 2-Crear Asigntura A valida 3-Creo Cursadas de la asignatura A 4-Agrego a A como competencia de P 4-Ejecutar prueba , pasando como parámetro P y A	Se da de baja al profesor en todas las cursadas que tienen como asignatura A y también se da de baja a A como competencia del profesor P	PASS
T4	Verificar el camino 4	Objeto Profesor,Objeto Asignatura	1-Crear Asignatura A valido 2-Crear Profesor P 3-Asignar a A como competencia de P 4-Ejecutar prueba	Se elimina la asignatura como competencia del profesor(No hay cursadas en el sistema)	PASS

Clase: Sistema

Método: void **agregarAlumnoEnCursada**(Alumno alumno, Cursada cursada) throws DatoInvalidoException, ClaveYaExistenteException

public void agregarAlumnoEnCursada(Alumno alumno, Cursada cursada) throws DatoInvalidoException, ClaveYaExistenteException

```
{
    if (!this.alumnoDisponible(alumno, cursada)) 1
        throw new DatoInvalidoException(alumno, "El alumno solicitado se encuentra ocupado en el horario de la cursada."); 2
    else
        cursada.altaAlumno(alumno); 3
} 4
```



Complejidad ciclomática: 4-4+2 = 2

C1: 1-2-4

C2: 1-3-4

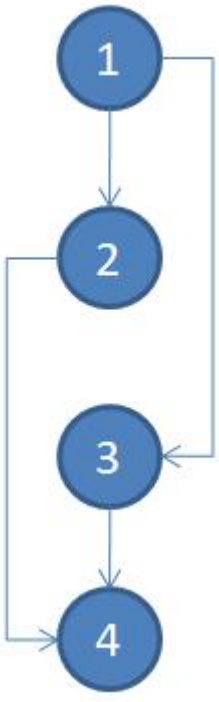
Id	Objetivo de la prueba	Datos de entrada	Procedimiento	Salida esperada	Resultado
T1	Verificar el camino 1	Estado de los objetos alumno y cursada	1- Crear un Alumno a 2-Crear una Asignatura asig1 y luego crear una Cursada c1 con los siguientes atributos: asignatura=asig1 período=01-2017; dia=LUN; horaInicio=10; horaFin=12 3-Crear una nueva Asignatura asig2 y una Cursada c2 asignándole los atributos: asignatura=asig2 período=1; día=Lunes; horaInicio=9; horaFin=11 4-Agregar a la Cursada c2 el Alumno a 5- Ejecutar prueba del método con los parámetros (a,c1)	DatoInvalidoException	PASS
T2	Verificar el camino 2	Estado de los objetos alumno y cursada	1- Crear un Alumno a 2-Crear una Asignatura asig y una Cursada c con los siguientes atributos: asignatura=asig período=01-2017; dia=LUN; horaInicio=10; horaFin=12 3- Ejecutar prueba del método con los parámetros (a,c)	Alumno agregado correctamente a la cursada	PASS

Clase: Sistema

Método: void **agregarProfesorEnCursada**(Profesor profesor, Cursada cursada) throws DatoInvalidoException, ClaveYaExistenteException

public void agregarProfesorEnCursada(Profesor profesor, Cursada cursada) throws DatoInvalidoException, ClaveYaExistenteException

```
{
    if (!this.profesorDisponible(profesor, cursada)) 1
        throw new DatoInvalidoException(profesor,
                "El profesor solicitado se encuentra ocupado en el horario de la cursada."); 2
    else
        cursada.altaProfesor(profesor); 3
} 4
```



Complejidad ciclomática: 4-4+2 = 2

C1: 1-2-4

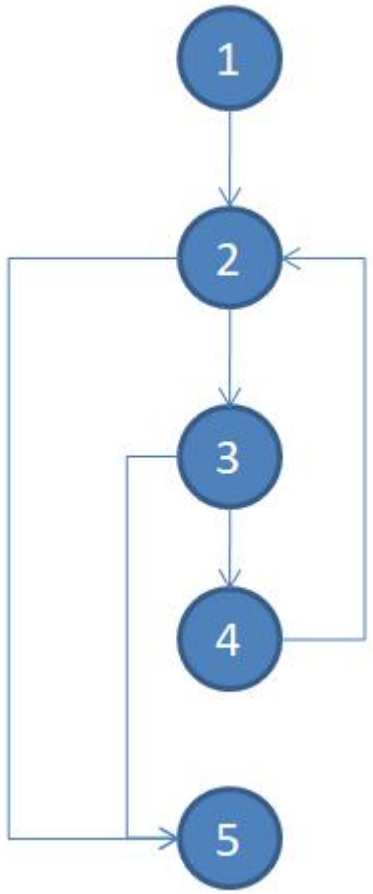
C2: 1-3-4

Id	Objetivo de la prueba	Datos de entrada	Procedimiento	Salida esperada	Resultado
T1	Verificar el camino 1	Estado de los objetos profesor y cursada	1- Crear un Profesor p 2-Crear una Cursada c1 de una Asignatura ya existente en el sistema, que tenga los atributos: período=1; dia=Lunes; horaInicio=10; horaFin=12 3-Crear una nueva Cursada c2 de otra Asignatura existente en el sistema, asignándole los atributos: período=1; día=Lunes; horaInicio=9; horaFin=11 4-Agregar la Cursada c1 y c2 a la competencia del profesor 4-Agregar a la cursada c2 el profesor p 5- Ejecutar prueba del método con los parámetros (p,c1)	DatoInvalidoException	PASS
T2	Verificar el camino 2	Estado de los objetos alumno y cursada	1- Crear un Profesor p 2-Crear una Cursada c de una Asignatura ya existente en el sistema, que tenga los atributos: período=1; dia=Lunes; horaInicio=10; horaFin=12 3-Agregar la Cursada c a la competencia del Profesor p 4- Ejecutar prueba del método con los parámetros (p,c)	Profesor agregado correctamente a la cursada c	PASS

Clase: Asignatura

Método: boolean **compruebaCorrelativas**(Alumno alumno)

```
public boolean compruebaCorrelativas(Alumno alumno)
  {
     Iterator<Asignatura> it = this.correlatividades.elementos();
     boolean ret = true; 1
     while (it.hasNext()  2 && ret 3)
          ret = alumno.asignaturaAprobada(it.next()); 4
     return ret; 5
  }
```



Complejidad ciclomática: 6 – 5 + 2 = 3

C1: 1-2-5

C2: 1-2-3-4-2-3-5

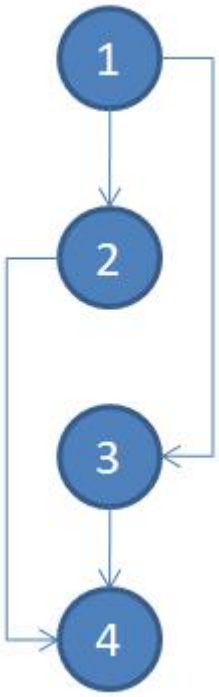
C3: 1-2-3-4-2-5

Id	Objetivo de la prueba	Datos de entrada	Procedimiento	Salida esperada	Resultado
T1	Verificar el camino 1	Estado del objeto alumno	1- Crear un Alumno a 2- Crear una Asignatura asig 3- Crear una Cursada de la Asignatura asig 4- Ejecutar prueba del método en relación a la Asignatura asig con el parámetro (a)	True	PASS
T2	Verificar el camino 2		1-Crear un Alumno a 2- Crear Asignaturas asig1, asig2, asig3 3- Establecer asig1 y asig 2 como correlativa de asig3 4- Ejecutar prueba del método en relacion a la Asignatura asig3 con el parámetro (a)	False	PASS
T3	Verificar el camino 3	Estado del objeto alumno	1-Crear un Alumno a 2- Crear Asignaturas asig1, asig2 3- Establecer asig1 y asig 2 como correlativa de asig3 4- Ejecutar prueba del método en relación a la Asignatura asig2 con el parámetro (a)	False	Test mal planteado

Clase: Sistema

Método: void **modificarAlumno**(Alumno alumno, Alumno modif)

```
public void modificarAlumno(Alumno alumno, Alumno modif)  throws DatoInvalidoException
  {
    if (!Persona.validaPersona(modif)) 1
        throw new DatoInvalidoException(modif, "Se detectaron parámetros inválidos al tratar de modificar."); 2
    else
        this.alumnos.modificarValor(alumno, modif); 3
  } 4
```



Complejidad ciclomática: 4-4+2 = 2

C1: 1-2-4

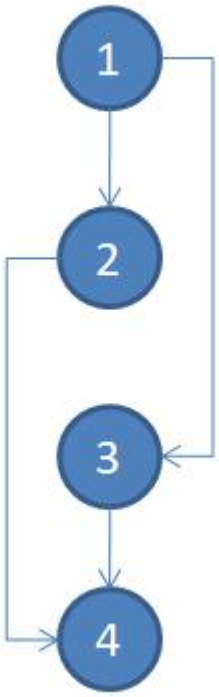
C2: 1-3-4

Id	Objetivo de la prueba	Datos de entrada	Procedimiento	Salida esperada	Resultado
T1	Verificar el camino 1	Estado del objeto alumno original y estado del objeto alumno modificado	1- Crear un Alumno a con los siguientes atributos (“Emanuel Ponce”, “Tejedor 123”, “manu@live.com.ar”) 2- Ejecutar la prueba pasándole como parámetro el Alumno a, y la instancia de un Alumno, modif, con los atributos (“Emanuel Nicolas Ponce”, “Tejedor 321”, “manulive.com.ar”)	DatoInvalidoException	PASS
T2	Verificar el camino 2	Estado del objeto alumno original y estado del objeto alumno modificado	1-Crear un Alumno a con los siguientes atributos (“Emanuel Ponce”, “Tejedor 123”, “manu@live.com.ar”) 2- Ejecutar la prueba pasándole como parámetro el Alumno a, y la instancia de un Alumno, modif, con los atributos (“Emanuel Nicolas Ponce”, “Tejedor 321”, “manu@live.com.ar”)	El alumno se modifica correctamente, por lo tanto posee el siguiente estado: apellidoNombre=”Emanuel Nicolas Ponce” domicilio=”Falucho 321” mail=“manu@live.com.ar”	PASS

Clase: Sistema

Método: modificarProfesor(Profesor profesor, Profesor modif)

```
public void modificarProfesor(Profesor profesor, Profesor modif) throws DatoInvalidoException
  {
    if (!Persona.validaPersona(modif)) 1
        throw new DatoInvalidoException(modif, "Se detectaron parámetros inválidos al tratar de modificar."); 2
    else
        this.profesores.modificarValor(profesor, modif); 3
  } 4
```



Complejidad ciclomática: 4-4+2 = 2

C1: 1-2-4

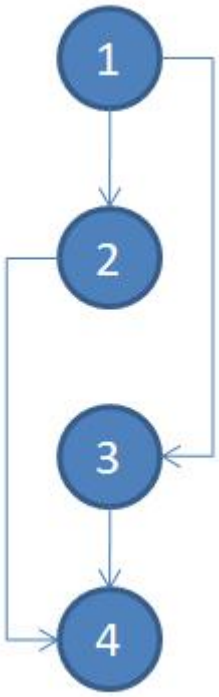
C2: 1-3-4

Id	Objetivo de la prueba	Datos de entrada	Procedimiento	Salida esperada	Resultado
T1	Verificar el camino 1	Estado del objeto profesor original y estado del objeto profesor modificado	1- Crear un Profesor p con los siguientes atributos (“Emanuel Ponce”, “Tejedor 123”, “manu@live.com.ar”, “4711949”) 2- Ejecutar la prueba pasándole como parámetro el Profesor p, y la instancia de un Profesor, modif, con los atributos (“Emanuel Nicolas Ponce”, “Tejedor 321”, “manulive.com.ar”, “4711949”)	DatoInvalidoException	PASS
T2	Verificar el camino 2	Estado del objeto profesor original y estado del objeto profesor modificado	1-Crear un Profesor p con los siguientes atributos (“Emanuel Ponce”, “Tejedor 123”, “manu@live.com.ar”,”4711949”) 2- Ejecutar la prueba pasándole como parámetro el Profesor p, y la instancia de un Profesor, modif, con los atributos (“Emanuel Nicolas Ponce”, “Tejedor 321”, manu@live.com.ar,”4711949”)	El profesor se modifica correctamente, por lo tanto posee el siguiente estado: apellidoNombre=”Emanuel Nicolas Ponce” domicilio=”Falucho 321” mail=”manu@live.com.ar” teléfono= “4711949”	PASS

Clase: Sistema

Método: void **modificarCursada**(Cursada cursada, Cursada modif) throws DatoInvalidoException

```
public void modificarCursada(Cursada cursada, Cursada modif) throws DatoInvalidoException
  {
    if (!Cursada.validaCursada(modif)) 1
        throw new DatoInvalidoException(modif, "Se detectaron parámetros inválidos al tratar de modificar."); 2
    else
        this.calendario.modificarValor(cursada, modif); 3
  } 4
```



Complejidad ciclomática: 4-4+2 = 2

C1: 1-2-4

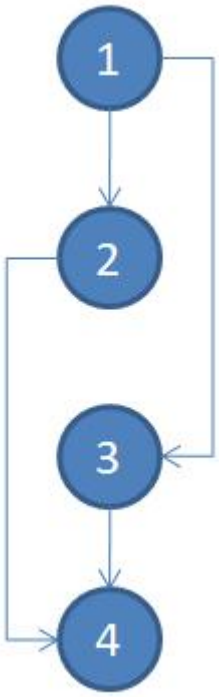
C2: 1-3-4

Id	Objetivo de la prueba	Datos de entrada	Procedimiento	Salida esperada	Resultado
T1	Verificar el camino 1	Estado del objeto cursada original y estado del objeto cursada modificada	1- Crear una Asignatura a 2- Crear una Cursada c con los siguientes atributos (a, “01-2017”,LUN, “10”,”12”) 2- Ejecutar la prueba pasándole como parámetro la Cursada c, y la instancia de una Cursada, modif, con los atributos (a, “01-2017”,LUN, “32”,”12”)	DatoInvalidoExceptio n	PASS
T2	Verificar el camino 2	Estado del objeto profesor original y estado del objeto profesor modificado	1- Crear una Asignatura a 2- Crear una Cursada c con los siguientes atributos (a, “01-2017”,LUN, “10”,”12”) 2- Ejecutar la prueba pasándole como parámetro la Cursada c, y la instancia de una Cursada, modif, con los atributos (a, “01-2017”,MIE, “10”,”13”)	La Cursada c se modifica correctamente pasando a poseer el siguiente estado: asignatura=a período=” 01-2017” día=MIE horaInicio=”10” horaFin=”13”	PASS

Clase: Sistema

Método: void **modificarAsignatura**(Asignatura asignatura, Asignatura modif)

```
public void modificarAsignatura(Asignatura asignatura, Asignatura modif) throws DatoInvalidoException
  {
    if (!Asignatura.validaAsignatura(modif)) 1
      throw new DatoInvalidoException(modif, "Se detectaron parámetros inválidos al tratar de modificar."); 2
    else
      this.planDeEstudio.modificarValor(asignatura, modif); 3
  } 4
```



Complejidad ciclomática: 4-4+2 = 2

C1: 1-2-4

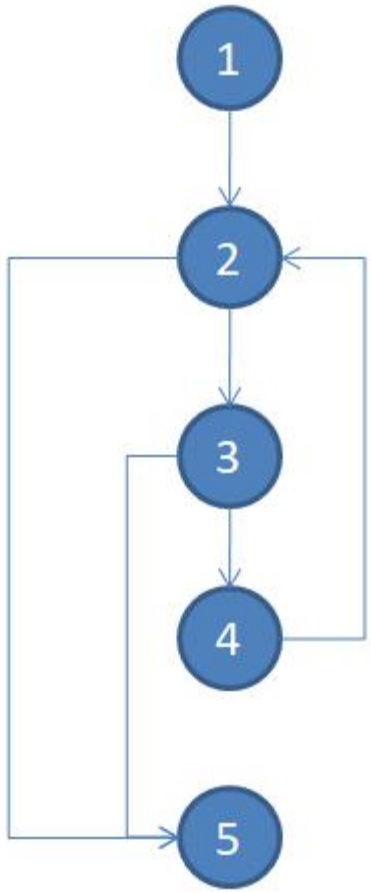
C2: 1-3-4

Id	Objetivo de la prueba	Datos de entrada	Procedimiento	Salida esperada	Resultado
T1	Verificar el camino 1	Estado del objeto asignatura original y estado del objeto asignatura modificada	1- Crear una Asignatura a con los siguientes atributos (“Analisis a”) 2- Ejecutar la prueba pasándole como parámetro la Asignatura a, y la instancia de una Asignatura, modif, con los atributos (“”)	DatoInvalidoExceptio n	PASS
T2	Verificar el camino 2	Estado del objeto asignatura original y estado del objeto asignatura modificada	1- Crear una Asignatura a con los siguientes atributos (“Analisis a”) 2- Ejecutar la prueba pasándole como parámetro la Asignatura a, y la instancia de una Asignatura, modif, con los atributos (“Analisis b”)	La Asignatura a se modifica correctamente pasando a poseer el siguiente estado: nombre=”Analisis b”	PASS

Clase: Sistema

Método: boolean **profesorDisponible**(Profesor profesor, Cursada cursada)

```
public boolean profesorDisponible(Profesor profesor, Cursada cursada)
  {
     boolean res = true;
     Cursada aux;
     Iterator<Cursada> it = this.calendario.elementosPorClavePrimaria(); 1
     while (it.hasNext() 2 && res 3)
     {
          aux = it.next();
          res = !aux.tieneProfesor(profesor) || !aux.hayColision(cursada); 4
     }
     return res; 5
  }
```



Complejidad ciclomática: 6 – 5 + 2 = 3

C1: 1-2-3-4-2-5

C2: 1-2-3-4-2-3-5

C3: 1-2-5

Id	Objetivo de la prueba	Datos de entrada	Procedimiento	Salida esperada	Resultado
T1	Verificar el camino 1	Estado del objeto profesor y estado del objeto cursada	1- Crear una Asignatura a 2- Crear una Cursada c con los siguientes atributos(a, “01-2017”,LUN, “10”,”12”) 3- Crear un Profesor p 4- Ejecutar la prueba con los parámetros (p,c)	True	PASS
T2	Verificar el camino 2	Estado del objeto asignatura original y estado del objeto asignatura modificada	1- Crear una Asignatura a1 y otra Asignatura a2 2- Crear una Cursada c1 con los siguientes atributos(a1, “01-2017”,LUN, “10”,”12”) 3- Crear una Cursada c2 con los siguientes	False	PASS

			atributos(a2, “01-2017”,LUN, “08”,”11”) 4- Crear un Profesor p 5-Agregar a la competencia del Profesor p la Asignatura a1 y la Asignatura a2 6- Agregar el Profesor p a la Cursada c1 6- Ejecutar la prueba con los parámetros (p,c2)		
T3	Verificar el camino 3	-	Impracticable	-	-