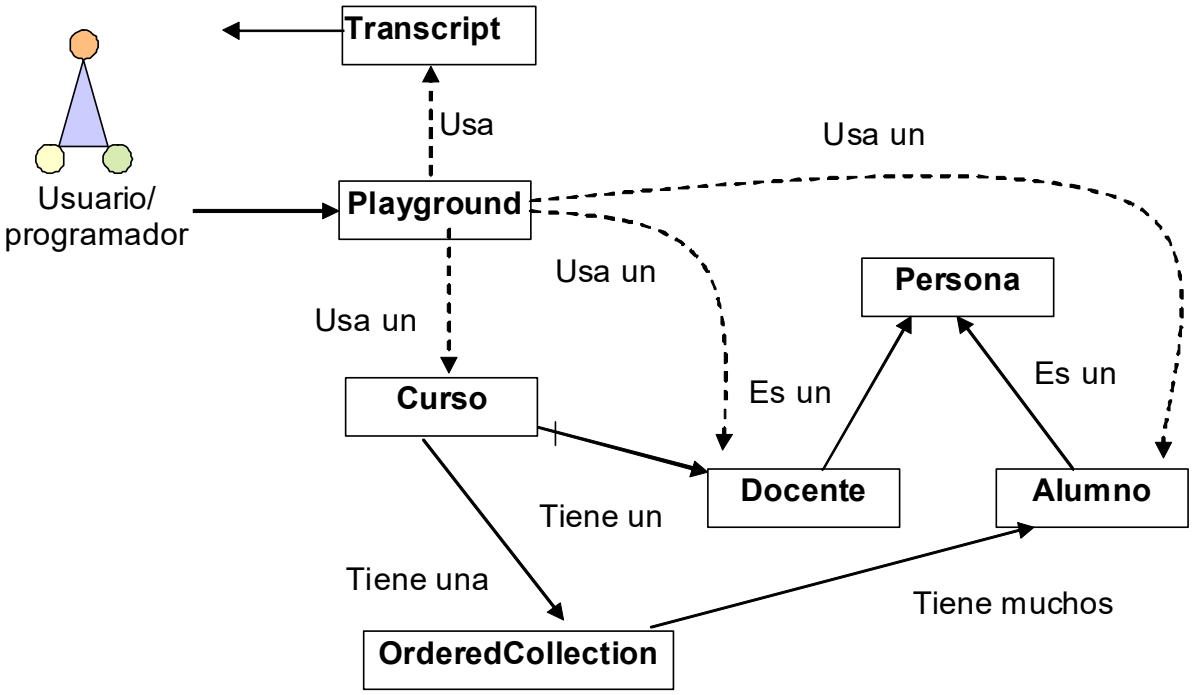
Caso de estudio – Clase 3
Utilización de Composición y Herencia

Enunciado

En base al siguiente modelo de clases:



Se desea implementar la operatoria básica de gestión de docentes y de alumnos de un curso, para ello se deberá realizar lo siguiente:

1. Implementar el modelo de clases correspondiente en Smalltalk.

2. Crear la instancia curso e invocar a sus métodos.

Diseño de clases

Persona: Atributos: dni y nombre.
Métodos: Inicializador, acceso, modificadores y asString.
Docente: Es una Persona.
Atributos: legajo y sueldo.
Métodos: Inicializador, acceso, modificadores y asString
Alumno: Es una Persona.
Atributos: legajo, nota1, nota2.
Métodos: Inicializador, acceso, modificadores y asString.
Un método de control que retorne el promedio del alumno.
Curso: Atributos: nombre, docente, alumnos.
Métodos: Inicializador, acceso, modificadores y asString, además los siguientes métodos para el manejo de los alumnos:

- listadoAlumnos: retorna el listado de alumnos.
- cantidadRegulares: retorna la cantidad de alumnos que tienen aprobadas sus dos notas.
- promedioTotal: retorna el promedio general del curso.

Resolución:

1. Diseño de clases

Implementación de clases

Clase Persona

```
Object subclass: #Persona
     instanceVariableNames: 'dni nombre'
     classVariableNames: ''
     category: 'ClaseHerencia'
```

Métodos de inicializacion

```
initialize
     "Inicializa la instancia de persona"
     dni := 0.
     nombre := ''.
```

Métodos de acceso

```
dni
     "Retorna el dni de la persona"
     ^dni.
nombre
     "Retorna el nombre de la persona "
     ^nombre.
```

```
asString
     "Retorna los datos de la persona en forma de cadena"
     ^('Dni : ', self dni asString, '- Nombre : ', self nombre asString).
```

Otra implementación

```
asString
     "Retorna los datos de la persona en forma de cadena"
     |retorno|
     retorno := 'Dni : ', self dni asString, '- Nombre : ', self nombre
asString.
     ^retorno.
```

Metodos modificadores

```
dni: unDni
     "Asigna un dni a la persona "
     dni:= unDni.
nombre: unNombre
     "Asigna unNombre a la persona "
     nombre:= unNombre.
```

Clase Docente

```
Persona subclass: #Docente
     instanceVariableNames: 'legajo sueldo'
     classVariableNames: ''
     category: 'ClaseHerencia'
```

Métodos de inicializacion

```
initialize
     "Inicializa la instancia de Docente"
     "Invoca a la instancia de la superclase"
     super initialize.
     legajo := 0.
     sueldo := 0.0.
```

Métodos de acceso

```
legajo
     "Retorna el legajo del docente "
     ^legajo.
sueldo
     "Retorna el sueldo del docente"
     ^sueldo.
```

```
asString
     "Retorna los datos del docente en forma de cadena"
     |retorno|
     retorno := super asString, '- Legajo : ', self legajo asString, '-
Sueldo : ', self sueldo asString, String cr.
     ^retorno.
```

Metodos modificadores

```
legajo: unLegajo
     "Asigna un unLegajo al docente "
     legajo := unLegajo.
sueldo: unSueldo
     "Asigna unSueldo al docente "
     sueldo:= unSueldo.
```

Clase Alumno

```
Persona subclass: #Alumno
     instanceVariableNames: 'legajo nota1 nota2'
     classVariableNames: ''
     category: 'ClaseHerencia'
```

Métodos de inicializacion

```
initialize
     "Inicializa la instancia de Alumno"
     "Invoca a la instancia de la superclase"
     super initialize.
     legajo := 0.
     nota1 := 0.
     nota2 := 0.
```

Métodos de acceso

```
legajo
     "Retorna el legajo del alumno"
     ^ legajo.
nota1
     "Retorna la nota1 del alumno"
     ^nota1.
nota2
     "Retorna la nota2 del alumno"
     ^nota2.
asString
     "Retorna los datos del alumno en forma de cadena"
     |retorno|
     retorno := super asString, '- Legajo : ', self legajo asString, '-
Nota1 : ', self nota1 asString, '- Nota2 : ', self nota2 asString, '-
Promedio : ', self promedio asString, String cr.
```

```
     ^retorno.
```

Metodos modificadores

```
legajo: unLegajo
     "Asigna unLegajo al alumno"
     legajo:= unLegajo.
nota1: unaNota1
     "Asigna unaNota1 al alumno"
     nota1:= unaNota1.
nota2: unaNota2
     "Asigna unaNota2 al alumno"
     nota2:= unaNota2.
```

Metodos de control

```
promedio
     "Retorna el promedio del alumno"
     |promedio|
     promedio := ((nota1+nota2) asFloat) /2.0.
     ^promedio.
```

Clase Curso

```
Object subclass: #Curso
     instanceVariableNames: 'nombre docente alumnos'
     classVariableNames: ''
     category: 'ClaseHerencia'
```

Métodos de inicializacion

```
initialize
     "Inicializa la instancia de curso"
     nombre := ''.
     docente := Docente new initialize.
     alumnos := OrderedCollection new.
```

Métodos de acceso

```
nombre
     "Retorna el nombre del curso "
     ^nombre.
docente
     "Retorna el docente del curso "
     ^docente.
alumnos
     "Retorna los alumnos "
     ^alumnos.
```

```
asString
     "Retorna los datos del curso en forma de cadena"
     |datosCurso|
     datosCurso:= String cr, ' El nombre del curso es: ', self nombre
asString.
datosCurso:= datosCurso, String cr, ' El nombre del docente es: ', self
docente asString nombre, String cr, 'Los alumnos son: ', self
listadoAlumnos.
     ^datosCurso.
```

Metodos modificadores

```
nombre: unNombre
     "Asigna unNombre al curso "
     nombre:= unNombre.

docente: unDocente
     "Asigna unDocente al curso "
     docente:= unDocente.

alumnos: unosAlumnos
     "Asigna unosAlumnos al curso "
     alumnos:= unosAlumnos.
```

Metodos de control

```
"Este método siempre se tiene que implementar cuando se utiliza
colecciones"
add: unAlumno
     "Asigna unAlumno a la coleccion del curso "
     self alumnos add:unAlumno.


listadoAlumnos
   "Retorna el listado de alumnos"
   | listado|
   listado:= ' Los alumnos cargados son: '.
   alumnos do: [ :unAlumno | listado:= listado, unAlumno asString. ].
   ^listado

cantidadRegulares
     "Retorna la cantidad de alumnos regulares del curso "
     |regulares|
      regulares :=0.

     "Recorre la colección y cuenta la cantidad de alumnos "
      alumnos do: [ :unAlumno |
              ( (unAlumno nota1>=4) &  (unAlumno nota2>=4) )
```

```
                    ifTrue: [regulares := regulares + 1] "fin del boque
if "
       ]. "fin del boque del do "

      ^regulares.
```

promedioTotal

```
     "Retorna el promedio general de los alumnos del curso "
     |promedio suma|
     suma :=0.0.

      alumnos do: [ :unAlumno | suma := suma + unAlumno promedio].
      promedio := (suma asFloat) / ((alumnos size) asFloat).

     ^promedio.
```

2. Utilización de instancias

1. Pruebas simples

1. "usa un alumno "

En la ventana de Playground ejecutar lo siguiente:

```
|unAlumno|

"Crea un objeto de tipo Alumno "
unAlumno:= Alumno new initialize.

"carga datos al alumno"
"invoca a todos los mensajes de acceso del alumno usando ;"

 unAlumno legajo: 33668; dni: 25458547; nombre: 'Juan'; nota1: 4;
nota2: 7.

"Muestra los datos al alumno "
Transcript show: 'El alumno cargado es: ', unAlumno asString; cr.
```

2. "usa un docente "

En la ventana de Playground ejecutar lo siguiente:

```
|unDocente|

"Crea un objeto de tipo Docente"
unDocente:= Docente new initialize.

"carga datos"
unDocente legajo: 33668; dni: 25458547; nombre: 'Juan'; sueldo:
25000.

"Muestra los datos"
Transcript show: 'El docente cargado es: ', unDocente asString; cr.
```

3. "usa un curso"

En la ventana de Playground ejecutar lo siguiente:

```
|curso unCurso unDocente alu1 alu2 alu3 |

"*****************************************"
"agrega un curso "
curso := Curso new initialize.

"carga el nombre dell curso"
curso nombre: '2k10'.
```

```
    "*******************************************"
    "crea un objeto de tipo Docente "
    unDocente:= Docente new initialize.

    "carga datos"
    unDocente legajo: 33668; dni: 25458547; nombre: 'Juan'; sueldo:
25000.

    "asigna unDocente al curso"
    curso docente: unDocente.

    "*******************************************"
    "Crea objetos de tipo Alumno y carga sus datos"
    alu1 := Alumno new initialize.
    alu1 legajo: 1;nombre: 'Lopez Maria'; nota1: 6; nota2: 8.

    alu2 := Alumno new initialize.
    alu2 legajo: 3;nombre: 'Perez Juan'; nota1: 9; nota2: 10.

    alu3 := Alumno new initialize.
    alu3 legajo: 2;nombre: 'Torrez Marcos'; nota1: 2; nota2: 2.


    "Carga los alumnos al curso"
    curso add: alu1.
    curso add: alu2.
    curso add: alu3.


    "*******************************************"
    "Muestra los datos del curso"
    Transcript show: String cr, 'Curso: ', curso asString.

    "Muestra los datos del docente a partir del curso"
    Transcript show: String cr, 'El docente del curso: ', (curso
docente) asString.

    "Muestra los datos de los alumnos a partir del curso"
    Transcript show: String cr, 'Los alumnos del curso: ', curso
listadoAlumnos.

    "Muestra los alumnos regulares a partir del curso"
    Transcript show: String cr, 'Alumnos regulares:  ',curso
cantidadRegulares asString.

    "Muestra el promedio de notas de los alumnos del curso"
    Transcript show: String cr, 'El promedio es: ', curso promedioTotal
asString.
```