El siguiente modelo de **Familia** muestra una única clase, llamada *Person,* y un tipo enumerado, llamado *Gender*, que como su propio nombre indica servirá para dar a cada instancia de la clase *Person* un género. A parte de este atributo, la clase *Person* cuenta con un atributo *age* de tipo *Integer* y con *name* de tipo *String*.

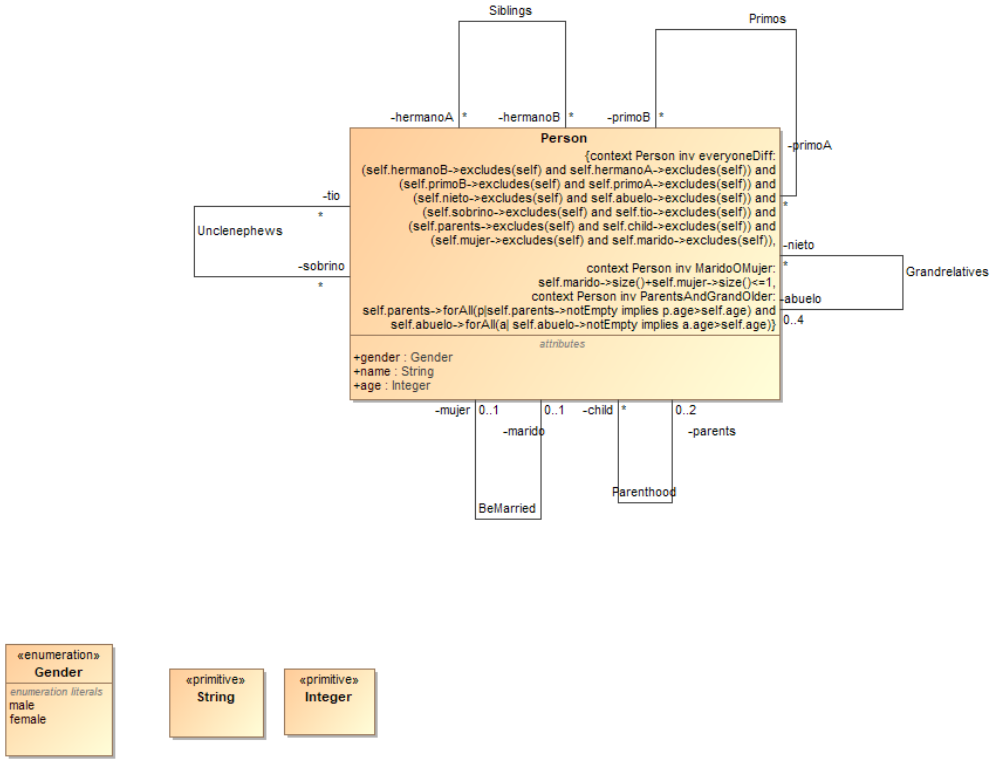


Imagen 1 : Modelo desarrollado en MagicDraw

Este modelo está pensado de forma que una persona pueda tener como máximo 2 padres y 4 abuelos, independientemente de su género. Además, se incluye la relación *BeMarried* que permite que una persona esté casada con otra, pero, aunque en la relación aparezcan roles de *marido* y *mujer*, estos solo servirán para desempeñar en la pareja el papel de cada uno de ellos, ya que no hay ningún constraint que prohíba el matrimonio entre personas del mismo sexo. También cuenta con las relaciones ***Siblings*** *(hermanos)*, ***Grandrelatives*** *(abuelo-nieto)*, ***Primos***, ***Unclenephews*** *(tio-sobrino)* y ***Parenthood*** *(padres-hijos)*.

De esta forma, a diferencia del modelo de Familia que vimos en nuestra primera sesión de laboratorio, permitimos construir una vista de toda la familia de un individuo, ya sea familia por parte de madre o de padre

El modelo consta de varios constraints de integridad que se muestran a continuación:

```
constraints

context Person inv everyoneDiff:
  (self.hermanoB->excludes(self) and self.hermanoA->excludes(self)) and
  (self.primoB->excludes(self) and self.primoA->excludes(self)) and
  (self.nieto->excludes(self) and self.abuelo->excludes(self)) and
  (self.sobrino->excludes(self) and self.tio->excludes(self)) and
  (self.parents->excludes(self) and self.child->excludes(self)) and
  (self.mujer->excludes(self) and self.marido->excludes(self))

context Person inv ParentsAndGrandOlder:
 self.parents->forAll(p|self.parents->notEmpty implies p.age>self.age) and
 self.abuelo->forAll(a| self.abuelo->notEmpty implies a.age>self.age)

context Person inv MaridoOMujer:
  self.marido->size()+self.mujer->size()<=1
```

Imagen 2: Constraints escritos en la aplicación USE

El **constraint** principal es ***everyoneDiff***, el cual se encarga de comprobar que una persona no es hermano de sí mismo, padre de sí mismo, nieto de sí mismo, etc. El segundo constraint es ***ParentsAndGrandOlder,*** cuyo propósito es comprobar que los padres y los abuelos de una persona son mayores que él mismo, utilizando el atributo *age*. Y, por último, pero no menos importante, tenemos la restricción ***MaridoOMujer***, que prohíbe que una persona sea marido de una y mujer a la vez de otra, permitiendo así que una persona solo pueda estar casada con otra persona.

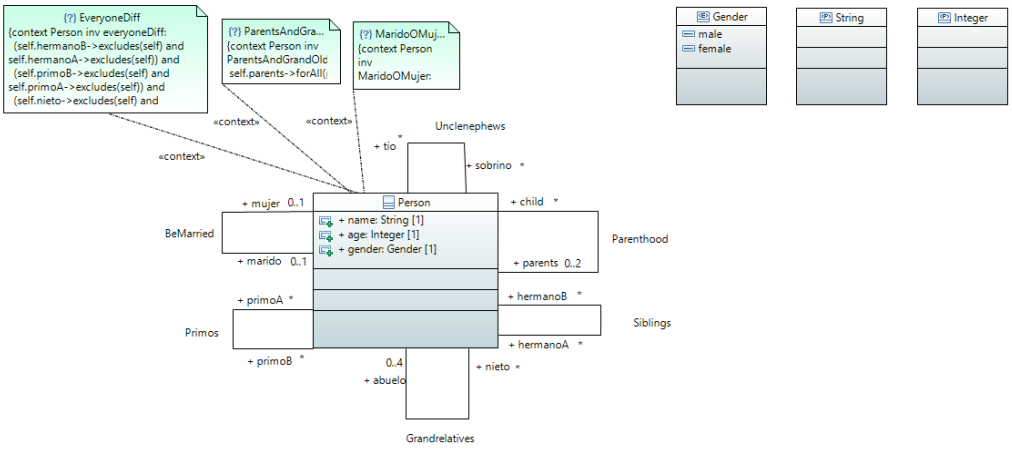


Imagen 3: El modelo desarrollado en Eclipse, usando Papyrus

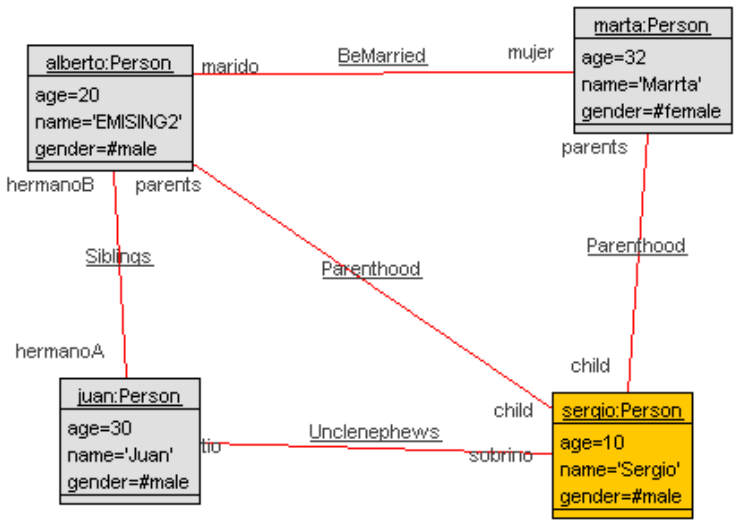
A continuación, probamos el modelo instanciándolo en USE. Para ello, consideraremos los siguientes casos:

CASO 1: Modelo instanciable, es decir, cumple los 3 requisitos de integridad.

```
!new Person('sergio')
!set sergio.age:=10
!set sergio.gender:=Gender::male
!set sergio.name:='Sergio'

!new Person('alberto')
!set alberto.age:=20
!set alberto.gender:=Gender::male
!set alberto.name:='EMISING2'

!new Person('juan')
!set juan.age:=30
!set juan.gender:=Gender::male
!set juan.name:='Juan'

!new Person('marta')
!set marta.age:=32
!set marta.gender:=Gender::female
!set marta.name:='Marrta'

!insert (alberto, sergio) into Parenthood
!insert (marta, sergio) into Parenthood
!insert (alberto, marta) into BeMarried
!insert (juan, alberto) into Siblings
!insert (juan, sergio) into Unclenephews
```

Código de ejecución 1

El resultado se muestra en la GUI de USE:



Y como paso final, comprobamos que la instancia que hemos creado cumple los requisitos de integridad del modelo:

```
use> check
checking structure...
checked structure in 4ms.
checking invariants...
checking invariant (1) `Person::MaridoOMujer': OK.
checking invariant (2) `Person::ParentsOlder': OK.
checking invariant (3) `Person::everyoneDiff': OK.
checked 3 invariants in 0.018s, 0 failures.
```

Resultado en consola 1

Por lo tanto, la instancia que hemos creado **cumple el modelo**.

```
!new Person('sergio')
!set sergio.age:=10
!set sergio.gender:=Gender::male
!set sergio.name:='Sergio'

!new Person('alberto')
!set alberto.age:=20
!set alberto.gender:=Gender::male
!set alberto.name:='EMISING2'

!new Person('juan')
!set juan.age:=30
!set juan.gender:=Gender::male
!set juan.name:='Juan'

!new Person('marta')
!set marta.age:=32
!set marta.gender:=Gender::female
!set marta.name:='Marrta'

!insert (alberto, alberto) into Parenthood
!insert (marta, sergio) into Parenthood
!insert (alberto, marta) into BeMarried
!insert (juan, alberto) into Siblings
!insert (juan, sergio) into Unclenephews
```

Código de ejecución 2

CASO 2: No cumple los constraints ***everyoneDiff*** y ***ParentsAndGrandOlder:***

Resultado en consola:

```
use> check
checking structure...
checked structure in 3ms.
checking invariants...
checking invariant (1) `Person::MaridoOMujer': OK.
checking invariant (2) `Person::ParentsOlder': FAILED.
  -> false : Boolean
checking invariant (3) `Person::everyoneDiff': FAILED.
  -> false : Boolean
checked 3 invariants in 0.014s, 2 failures.
```

Resultado en consola 2

CASO 3: Incumple el requisito ***MaridoOMujer***.

```
!new Person('sergio')
!set sergio.age:=10
!set sergio.gender:=Gender::male
!set sergio.name:='Sergio'

!new Person('alberto')
!set alberto.age:=20
!set alberto.gender:=Gender::male
!set alberto.name:='EMISING2'

!new Person('juan')
!set juan.age:=30
!set juan.gender:=Gender::male
!set juan.name:='Juan'

!new Person('marta')
!set marta.age:=32
!set marta.gender:=Gender::female
!set marta.name:='Marrta'

!insert (alberto, sergio) into Parenthooc
!insert (marta, sergio) into Parenthood
!insert (alberto, marta) into BeMarried
!insert (juan, alberto) into Siblings
!insert (juan, sergio) into Unclenephews
!insert (marta, juan) into BeMarried
```

Código de ejecución 3

Resultado en consola:

```
use> check
checking structure...
checked structure in 4ms.
checking invariants...
checking invariant (1) `Person::MaridoOMujer': FAILED.
  -> false : Boolean
checking invariant (2) `Person::ParentsOlder': OK.
checking invariant (3) `Person::everyoneDiff': OK.
checked 3 invariants in 0.016s, 1 failure.
```

Resultado en consola 3