

Instance and Class Properties



In this section we are extending the concept of instances. The concept is used consistently on the object as well as class level. Explanations of **instance and class properties** are covered in this section.

Instance Properties

So far, objects have been described by attributes or properties, which have been put into (declared in) classes. To be exact, these properties are called **instance properties**, because each instance (or object) of the class has these properties and they can be changed individually. The following diagrams show the instance properties for the Sloopies Dima, Nina and Eddy.

Property	Type	Value
hair	Colour	yellow
eyes	Colour	green
age	Number	9
jumper	Colour	blue
shoes	Colour	yellow
belt	Yes/No	Yes



Figure 1: Instance properties of Dima

Property	Type	Value
hair	Colour	green
eyes	Colour	blue
Age	Number	10
jumper	Colour	yellow
shoes	Colour	pink
belt	Yes/No	No



Figure 2: Instance properties of Nina

Property	Type	Value
hair	Colour	blue
eyes	Colour	blue
Age	Number	12
jumper	Colour	red
shoes	Colour	yellow
belt	Yes/No	Yes




Figure 3: Instance properties of Eddy

Class Properties

Additionally, a class by itself can have properties, so called **class properties**. Each property only exists once in each class. An example of a class property is the number of instances within the class. We will extend our *Sloopy* class by a **class property** named *Counter* to keep track of the number of Sloopies. That is, every time a new Sloopy is created (instantiated) the counter will be incremented by 1; every time a Sloopy is deleted, *Counter* will be decremented by 1.


Property	Type	Value	
hair	Colour	yellow	
eyes	Colour	green	
Age	Number	9	
jumper	Colour	blue	
shoes	Colour	yellow	
belt	Yes/No	Yes	
<i>Counter</i>	Number	1	
Property	Type	Value	
hair	Colour	green	
eyes	Colour	blue	
age	Number	10	
jumper	Colour	yellow	
shoes	Colour	pink	
belt	Yes/No	No	
<i>Counter</i>	Number	2	
Property	Type	Value	
hair	Colour	blue	
eyes	Colour	blue	
age	Number	12	
jumper	Colour	red	
shoes	Colour	yellow	
belt	Yes/No	Yes	
<i>Counter</i>	Number	3	

Figure 4: Class property *Counter*

It is common practise in UML to start **instance properties** in lower case and **class properties** in upper case. *Counter* has been allocated the type *Long*, which is a number that can represent a large number of counters.

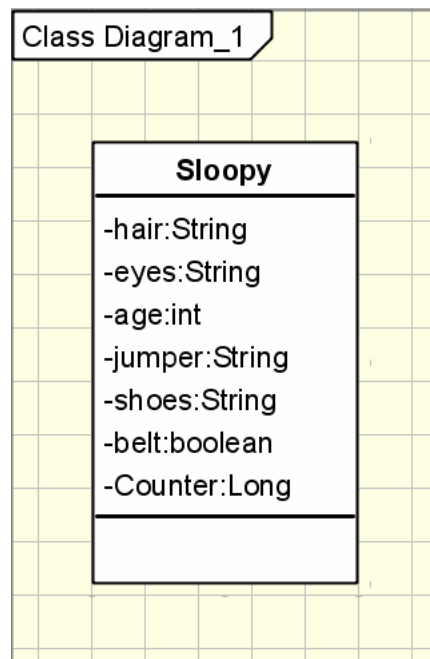


Figure 5: UML class with class property *Counter* of type *Long*

Figure 6 shows a snapshot of all the **class properties** and **instance properties** of our Sloopies at a given moment. Notice the *Counter* property is 3 for all the instances (objects). This shows that a class property will remain the same for all Instances of the class.

Property	Type	Value		Property	Type	Value		Property	Type	Value	
Hair	Colour	yellow		hair	Colour	green		hair	Colour	blue	
Eyes	Colour	green		eyes	Colour	blue		eyes	Colour	blue	
age	Number	9		age	Number	10		age	Number	12	
jumper	Colour	blue		jumper	Colour	yellow		jumper	Colour	red	
shoes	Colour	yellow		shoes	Colour	pink		shoes	Colour	yellow	
belt	Yes/No	Yes		belt	Yes/No	No		belt	Yes/No	Yes	
Counter	Number	3		Counter	Number	3		Counter	Number	3	

Figure 6: Snapshot of Sloopy class and instances properties

Dot Notation

In formal approaches, adding values to objects is represented by the so-called **dot notation**, where it can be seen which object belongs to which class and which property belongs to which instance or class.

```
Sloopy.Eddy.hair := blue  
Sloopy.Eddy.eyes := blue  
Sloopy.Eddy.age := 12  
Sloopy.Eddy.jumper := red  
Sloopy.Eddy.shoes := yellow  
Sloopy.Eddy.belt := true  
Sloopy.Eddy.Counter := 1
```



```
Sloopy.Nina.hair := green  
Sloopy.Nina.eyes := blue  
Sloopy.Nina.age := 10  
Sloopy.Nina.jumper := yellow  
Sloopy.Nina.shoes := pink  
Sloopy.Nina.belt := false  
Sloopy.Nina.Counter := 2
```



You should now be familiar with the basic concepts of structural elements of object-oriented modelling. These include objects, classes and instances.