4 ATTRIBUTES AND OPERATIONS

Terry Marris 8 April 2001

4.1 OBJECTIVES

By the end of this lesson the student should be able to

- understand the difference between attributes and operations
- appreciate the purpose of constructors, setting and getting operations
- understand the concept of information hiding

4.2 PRE-REQUISITES

The student should be familiar with class diagrams, strings and the non-object number types *int* and *double* (chapters one, two and three).

4.3 PREVIEW

We begin to put some detail into our class diagrams. We look at attributes and operations, and the private and public scope modifiers.

4.4 ATTRIBUTES

We start with a familiar class: Person.

A Person has a name, an age and a height.

A Person knows all about itself. It can tell you their name, their age and their height. And they can change their name, age and height.

Name, age and height are all examples of Person attributes.

An attribute is derived from what an object *has*. For example, a book Copy object *has* an author, a title and a publisher. So the Copy class would have attributes author, title and publisher.

Attributes are often (but not always) small simple classes such as strings, and non-object values such as *int* and *double*.

An attribute is always singular, e.g. a *name*, not names.

Some examples of attribute specifications are shown below in Figure 6.1.

attribute	description
name:String	name is a string
age:int	age is an int
height:double	height is a double

Figure 4.1 *Attribute Specifications*

4.5 OPERATIONS

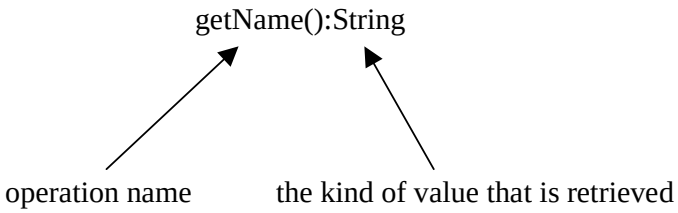
Operations are the processes that a class carries out (usually) on its attributes.

4.5.1 GETTING OPERATIONS

A getting operation tells you what value is stored in an attribute. The operation *getName()* might tell you the value stored in a *name* attribute.

operation	description
getName():String	retrieves the value stored in a name attribute (a string)
getAge():int	retrieves the value stored in a budget attribute (an int)
getHeight():double	retrieves the value stored in a height attribute (a double)

Figure 4.2 *Getting Operations*

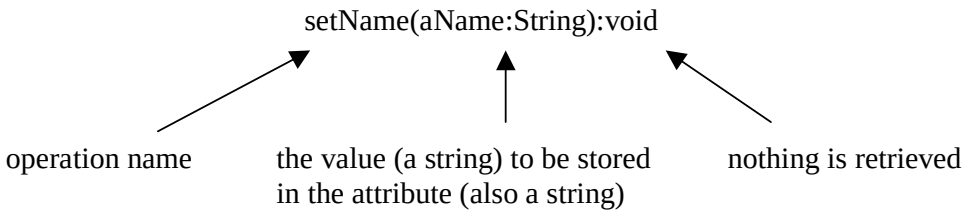


4.5.2 SETTING OPERATIONS

A setting operation changes the value stored in an attribute. The *setAge(anAge:int)* operation might put the *int* value *anAge* into an *age* attribute.

operation	description
setName(aName:String):void	stores aName in a name attribute
setAge(anAge:int):void	stores anAge in an age attribute
setHeight(aHeight:double):void	stores aHeight in a height attribute

Figure 4.3 *Setting Operations*

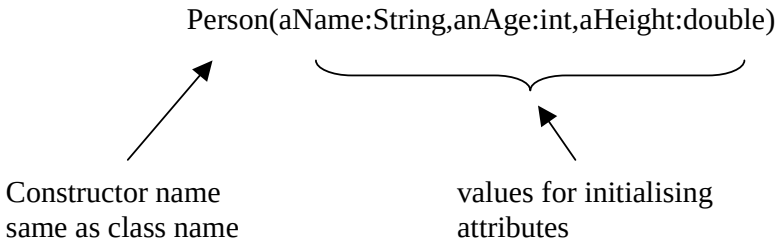


4.5.3 CONSTRUCTOR OPERATIONS

A constructor operation initialises attributes, that is, gives attributes their initial or starting values.

operation	description
Person(aName:String)	initialises a name attribute with aName
Person(aName:String,anAge:int,aHeight:double)	initialise a name attribute with aName, initialises an age attribute with anAge, initialises a height attribute with aHeight.

Figure 4.4 *Constructor Operations*



Operation names differ from attribute names in that an operation has a pair of brackets following its name. So, *area* would indicate an attribute, but *area()* would indicate an operation of some kind.

4.6 CLASS DIAGRAMS

We can show attributes and operations in a class diagram.

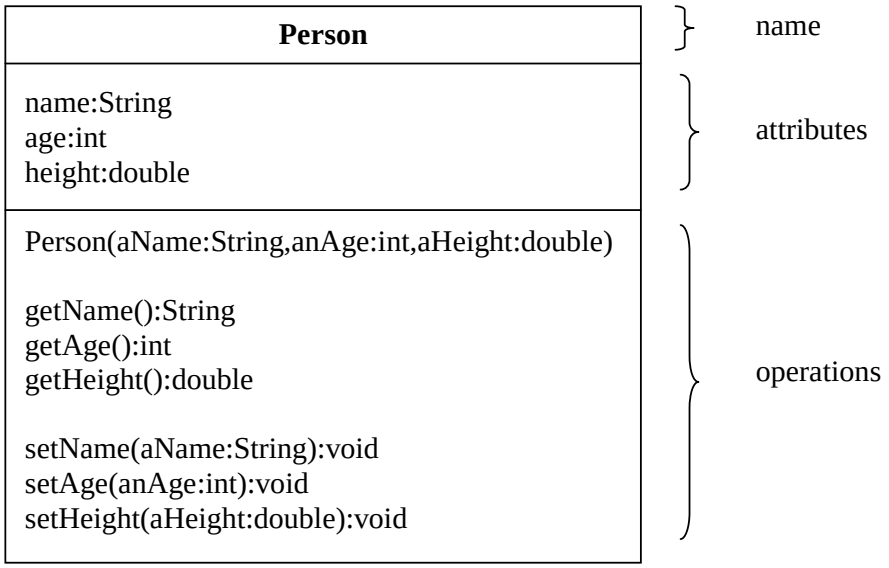


Figure 4.5 *Class Diagram showing Attributes and Operations*

A class diagram has three sections.

The section at the top is for the name of the class.

The middle section is for attributes.

The section at the bottom is for operations.

We are free to put as much or as little detail as we like into our class diagrams, whatever we find to be useful. However, we do aim to be consistent.

Since we can reasonably assume that every attribute has a getting and a setting operation associated with it, we do not need to show the setting and getting operations on a class diagram if we do not want to. We do whatever is clear to our readers and to ourselves.

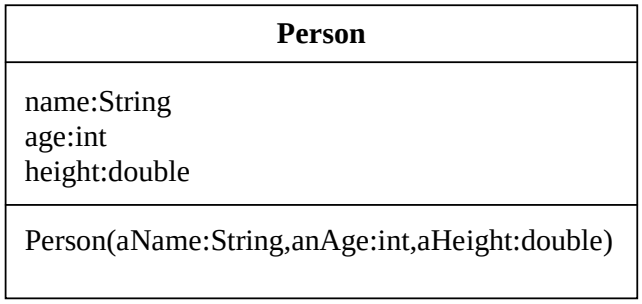


Figure 4.6 *Class Diagram showing Attributes and a Constructor*

We infer the getting and setting operations from the given attributes.

4.7 INFORMATION HIDING

We make all attributes *private*. A *private* attribute can only be accessed by the operations from its own class.

In general (meaning there are exceptions) we make all operations *public*. Any object from any class can use a *public* operation.

This means that the users of our classes (known as clients) cannot have direct access to our attributes. They can only access them through the operations that we provide. This restricted access helps prevent the data stored in our attributes from being changed in ways we did not expect or want.

In fact, our clients do not need to know about our attributes; they only need to know about the operations (or services) we provide. This is known as information hiding - giving clients access to data only on a need to know basis.

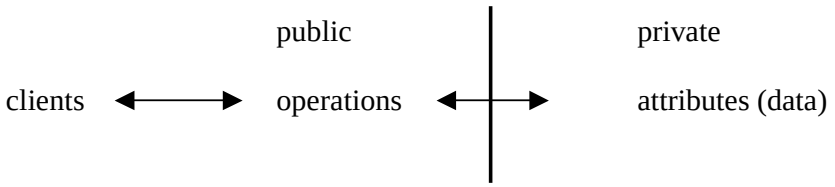


Figure 4.7 *Private data reaches clients through public operations*

In a class diagram

an attribute is specified as private by placing a minus sign before it

an operation is specified as public by placing a plus sign before it.

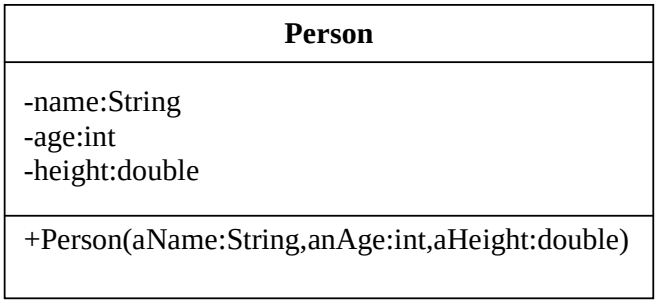
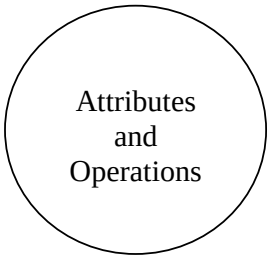


Figure 4.8 *Class Diagram showing Attributes and a Constructor*

4.8 FURTHER READING

FOWLER M & SCOTT K *UML Distilled* pp 99

4.9 REVIEW



Attribute

what an object has

Operation

some action performed on an attribute

Class diagram

name

attributes

operations

constructors

initialises attributes

setting

stores values in attributes

getting

retrieves values from attributes

private attributes

accessed only by its class operations

public operations

used by any object from any class

4.10 EXERCISES

1 Explain, with the aid of examples, what is meant by each of the following terms:

- **(a)** attributes
- **(b)** operations
- **(c)** private
- **(d)** public
- **(e)** constructor
- **(f)** setting operations
- **(g)** getting operations

2 Represent each of the following scenarios in a class diagram.

(a) A bus has a registration number and a seating capacity.

(b) A bus driver has a name, a number and an hourly rate of pay.

(c) A music track has a composer, a title, a performer and a duration.

(d) A circle has a centre represented by some co-ordinate position (x, y) and a radius.

(e) A window (on a computer screen) has a title, a width, a height and a position represented by the (x, y) co-ordinates of its top left hand corner.