# 5. Data items and types of data items

## Description

In this chapter, we describe data-items that may represent references to objects, primitive values, string values, and composite values.

As mentioned in previous chapters, objects may have attributes in the form of data-items that represent properties characterizing the corresponding phenomena. For the `Account` example, we have seen examples of data-items such as `owner`, `balance`, and `interestRate`:

```
class Account(owner: ref Customer):
   balance: var float
   interestRate: var float
   -"-
```

There is a fundamental difference between the data-items `owner` and `balance`. While `owner` is a *reference* to the object representing the owner of the account (represented by an object of class `Customer`), `balance` represents a property that is given by a float *value*. We return to this in section .
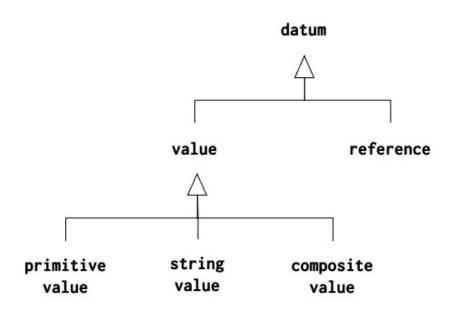
Some values are denoted *primitive values*, and their types are often predefined. `balance` and `interestRate` are examples of data-items holding primitive values, in this case of type `float`. We have also seen examples of primitive values of type `integer` and `boolean`. We summarise these primitive values below.

A phenomenon may also be characterised by values that are composite values constructed from primitive values or other composite values. Examples of this is a date, which consist of a year, month and day all of type integer. Other examples are points, complex numbers, GPS-positions, etc. The type of a composite value is called a *composite value type* or just *composite type*. We describe composite values and types in section .

A `String` is another example of a value type. The name attribute of `Customer` was in an early version the class of type `String`. We describe `String` in section .

We use the term *datum* to cover values as well as references. Data-items in general may hold datums and a datum may thus be a value or reference.

The tree in the figure shows a so-called classification hierarchy. The nodes in the tree are concepts and the arrows illustrates the relationships between the concepts. As said, a datum may be value or reference. A value in turn may be a primitive value, a string value or a composite value. We further discuss classification hierarchies in section  and we show how to represent such hierarchies in a program.

datum

value                    reference

primitive          string          composite
value              value           value

Datum hierarchy