

Aggregation

- Purpose: To understand the special forms of association called aggregation and composition

What is aggregation?

Aggregation is not a concept unique to object-oriented programming languages. Any language that supports record-like structures supports aggregation.

Key concept

Aggregation:

Aggregation is a special form of association between objects with the meaning of a whole/part hierarchy together with the ability to navigate from the whole or aggregate to its parts. If and only if there exists a whole/part relationship between two objects, e.g. book and page, can an aggregate relationship exist between objects of their corresponding classes, TBookClass and TPageClass.

Aggregation is a special form of association with the meaning of a whole/part hierarchy together with the ability to navigate from the whole or aggregate to its parts.

For example, a book has pages. We can think of a book as the whole and the pages as the parts.

A book identified with the ID 36 could be made up of 500 pages with each page uniquely numbered.

The book's title might be 'Intro to OOP by L. P. Partley'.

The pages are bound together and a cover attached to make the book.

In a simplified scenario, we might define a class TBook for all book objects as shown in Table 24.1.

Methods have been omitted from TBook for convenience.

```
TBook = Class
    BookID : 1..2000;
    Title : String;
    PageRef : Array[1..500] Of TPage;
End;
```

Table 24.1 TBook class definition

```
TPage = Class
    PageNo : Integer;
    PageContent : String;
End;
```

Table 24.2 TPage class definition

```
aBook := TBook.Create;
aBook.BookID := 36;
For i := 1 To 500 Do
    Begin
        aBook.PageRef[i] := TPage.Create;
        aBook.PageRef[i].PageNo := i;
    End;
```

Table 24.3 Creating book and page objects

Similarly, we might define a class TPage for all the page objects which make up a book as shown in Table 24.2.

Methods have been omitted from TPage for convenience.

We use an object constructor method

Create to create an instance of the TBook class and assign the reference to a variable aBook of type TBook.

We can then create 500 page objects and assign their references as shown in Table 24.3.

We navigate from the whole to the part as follows

```
Writeln(aBook.PageRef[2].PageNo);
```

This Writeln statement accesses aBook, the whole, then an attribute of aBook, called PageRef. This attribute is an array which can contain up to 500 references to TPage objects. The indexed reference PageRef[2] references the second page object, a part of the whole. This has an attribute PageNo. It is this which is displayed by Writeln.

The book object may be destroyed, i.e. the memory occupied by the object released, by using a user-defined **destructor**, `Destroy`, with body code as shown in [Table 24.4](#).

The code first destroys all the page objects before finally destroying the book object. The user-defined destructor `Destroy` releases the memory of the object to which it is applied.

If the book object was a tree in a forest then its leaves are destroyed first followed by the tree itself.

The detail shown in [Table 24.4](#) can be hidden within the user-defined destructor so that calling `aBook.Destroy` destroys everything. By analogy, the whole tree is destroyed, leaves, trunk, branches, etc, by the action of destruction.

```
For i := 1 To 500
  Do aBook.PageRef[i].Free;
aBook.Free;
```

Table 24.4 Releasing the memory occupied by the book and page objects

Questions

- 1 Assuming that a book object has been created with 500 pages, each numbered consecutively starting from 1, what is displayed by `Writeln` if the following code could be executed?

```
Writeln(aBook.PageRef[36].PageNo);
```

Composition

The book example is actually an example of a stronger form of aggregation called **composition**.

A whole/part relationship or association in which the aggregate object cannot exist without its components. A book is not a book without its pages otherwise it would just be a book cover.

Suppose we changed the `TBook` class so that its attributes are private and control access to these attributes via methods such as `TurnPage`, `ShowPage`. Our class definition might now resemble that shown in [Table 24.5](#).

All page objects are now hidden behind access methods. The consequence is that each page is **associated** with just one book object because there is no easy way to copy a page reference so the object it references can be *associated* at the same time with another book object.

```
TBook = Class
  Private
    BookID : 1..2000;
    Title : String;
    PageRef : Array[1..500] Of TPage;
  Public
    Procedure TurnPage;
    Procedure ShowPage;
    ...
End;
```

Table 24.5 Applying data encapsulation to TBook

Composition is characterised by the following

1. It is a whole/part relationship or association, i.e. an aggregation
2. The aggregate object cannot exist without its component objects
3. An object is composed of other objects in a fixed relationship
4. The components can only belong to a single composition, i.e. cannot be shared with another aggregate object at the same time

It is not a defining characteristic of composition to say that the runtime lifetime of the part is the same as the whole, i.e. destroy the whole and the parts are destroyed too; however, it is usually the case. An example of when it is not is a bicycle wheel which could be removed and used with another bicycle. The bicycle without a wheel is no longer a bicycle.

The main characteristics are that the aggregate object cannot exist without its component objects and that these are fixed and not shared with another aggregate object at the same time.

Composition: Whole/part association in which the aggregate object cannot exist without its component objects and the components are fixed and can only belong to a single composition.

Key concept

Composition:

A whole/part relationship or association in which the aggregate object cannot exist without its components. A composition hierarchy defines how an object is composed of other objects in a fixed relationship. The components can only belong to a single composition

Aggregation

When the aggregation is weaker than composition we just call it **aggregation**. This is a whole/part relationship or association in which none of the part objects are essential for the whole to exist. This implies that the part is also capable of an independent existence and the whole/part association is variable - part objects 'come and go'. For example, a student object may be associated with both a **TutorGroup** object (a pastoral group in a school) and a **SportsTeam** object but may switch in and out. A **TutorGroup** object has students and a **SportsTeam** object has students so in both cases the relationship is whole/part but the relationship is not a fixed one and the aggregate/whole object in each case can exist without its component objects. The component objects can be added later from a 'pool' of these which are shared out amongst the 'whole' objects. In a clock example, batteries may be provided from a 'pool of batteries' and batteries may be added or removed from a clock and even used in another clock without affecting the existence of the clock which would be the case, for example if the clock face was removed. The whole/part relationship of battery objects is thus aggregation, not composition.

Aggregation:

In aggregation the whole/part association is variable.

However, the concept of an aggregation which isn't composition doesn't mean much in practice so it is often just called an association. Therefore, it is not essential to use aggregation, but it can help us to understand and give added meaning to a model.

A simple test can be applied to discover if the association or relationship between two objects is aggregation. It applies to both composition and (non-composition) aggregation.

The test is called the **"has/has-a"** or **"is part of"** or **"contains"** test, e.g. **a book has a number of pages** indicates that the relationship is aggregation.

Key concept

Aggregation:

It is a whole/part relationship or association in which an object is composed of a variable set of component objects, e.g. a **SportsTeam** object composed of student objects, but the ones used are not always the same. Part objects are not essential for the whole object to exist, e.g. an empty car boot whole object. The part is also capable of an independent existence, e.g. the contents of a car boot are potential part objects. It is possible for a part to be associated with more than one whole (not necessarily at the same time), e.g. a student object can be a part of a sports team object and a tutor group object.

Windows form composition application

Figure 24.1 shows a Windows form application which adds two integers and displays the result.

The form object `FormAdd` is of type `TFormAdd`.

`TFormAdd` is a class with seven attributes and one method which performs the addition.

Table 24.7 shows the Delphi unit which contains the class definition `TFormAdd`, and Table 24.6 shows the program which uses the class `TFormAdd` to create a form object which is referenced by the reference variable `FormAdd`. This form object has a `TEdit` attribute called `EditFirst`.

`EditFirst` is an object of type `TEdit`.

`TEdit` is a class supplied by the Delphi programming language.

The "has-a" test tells us that the relationship or

association is

aggregation,

and since the

`EditFirst`

object has a fixed

relationship with

the form object, this

is aggregation of the

composition type

- see class diagram

Figure 24.2. The

same test is passed

by the other objects,

`AddBtn` and so on.

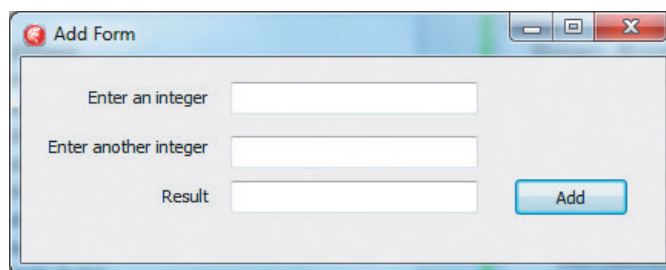


Fig. 24.1 Windows form object containing other objects

```
Program AddProject;
Uses
  Vcl.Forms, AddUnit in 'AddUnit.pas' {FormAdd};
{$R *.res}
Begin
  Application.Initialize;
  Application.MainFormOnTaskbar := True;
  Application.CreateForm(TFormAdd, FormAdd);
  Application.Run;
End.
```

Table 24.6 Delphi program which creates form

```
Unit AddUnit;
Interface
Uses
  Winapi.Windows, Winapi.Messages, System.SysUtils, System.Variants,
  System.Classes, Vcl.Graphics, Vcl.Controls, Vcl.Forms, Vcl.Dialogs, Vcl.StdCtrls;

Type
  TFormAdd = Class(TForm)
    AddBtn: TButton;
    EditFirst: TEdit;
    LabelEnterFirst: TLabel;
    EditSecond: TEdit;
    LabelEnterSecond: TLabel;
    Result: TEdit;
    LabelResult: TLabel;
    Procedure AddBtnClick(Sender: TObject);
  End;

Var
  FormAdd: TFormAdd;
Implementation
{$R *.dfm}
Procedure TFormAdd.AddBtnClick(Sender: TObject);
Begin
  Result.Text := IntToStr(StrToInt(EditFirst.Text) + StrToInt(EditSecond.Text));
End;
End.
```

Table 24.7 Delphi unit for a windows form containing seven objects

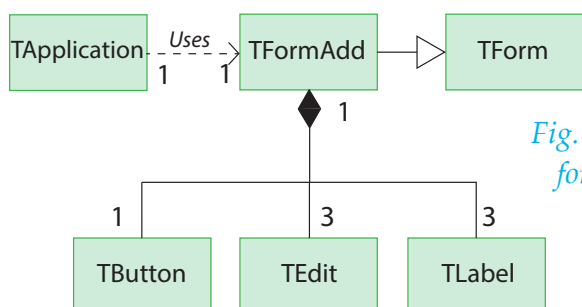


Fig. 24.2 Class diagram for Windows form application - see Chapter 27

Further information

There are relationships or associations which are not aggregation or inheritance - see Chapter 27. These associations denote semantic dependency among otherwise unrelated classes, such as between "snakes" and "long grass" – snakes hide in long grass.

These are just referred to as associations. For example, the type of association called *dependency* is a "uses"

relationship. This is used to specify the classes "used" by the contents of the class concerned. This is usually taken to mean that the object class acts upon another class, or needs information contained within another class.

For example, a "snake" can move on the "ground" and therefore can be said to have a dependence relationship with "ground". However, since a "snake" is not "ground", it is not an inheritance relationship, and a "snake" does not contain "ground" (i.e. a snake has a ground does not make sense), a "snake" does not have an aggregation relationship with "ground".

Questions

- 2 A bicycle consists of a frame, wheels, tyres, a gear train, pedals, saddle, handlebars, brakes and brake levers, and a gear lever.
What is the relationship between a bicycle object and its parts?
- 3 A game of chess uses a chess board and chess pieces. A chess board consists of black and white squares.
What is the relationship between chess board and its black and white squares?
- 4 A digital clock is a kind of clock and an analogue clock is also a kind of clock. An analogue clock consists of a clock face, clock hands, a clock mechanism.
What is the relationship between
 - (a) Clock and digital watch?
 - (b) Clock and analogue watch?
 - (c) An analogue watch and clock face, clock hands, clock mechanism?
 - (d) A digital clock and a battery used to power the clock?