

Class diagrams

Purpose: To learn how to create class diagrams

Single inheritance, association, aggregation and composition were covered in a practical way in [Chapter 22](#) and [Chapter 24](#).

A class diagram is used to show the relationships/associations between classes.

[Figure 27.1](#) shows how single inheritance is represented in a class diagram consisting of two classes A and B. Class B is the derived class or subclass which inherits from class A. Each class is drawn as a labelled rectangle.

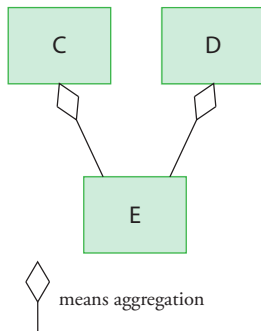
[Figure 27.2](#) shows how aggregation (variable aggregation) is represented in a class diagram consisting of three classes C, D and E.

In the whole/part association known as aggregation, Classes C and D are the *whole* and class E is the variable set of component objects *part*.

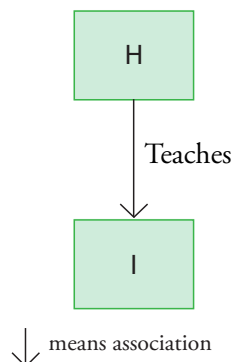
[Figure 27.3](#) shows how composition (fixed aggregation) is represented in a class diagram consisting of two classes F and G. In the whole/part association known as composition, Class F is the *whole* and class G is the *part*. The aggregate object (class F object) cannot exist without its component objects drawn from class G.

[Figure 27.4](#) shows how association is represented in a class diagram consisting of two classes H and I. The direction of the arrow indicates that the association is from H to I, e.g. "teacher teaches student" where "teacher" corresponds to H and "student" corresponds to I. The association is "teaches". [Figure 27.5](#) shows how association of the dependency kind is represented in a class diagram. The direction of the arrow indicates that the association is from J to K, e.g. "snake uses ground" where "snake" corresponds to J and "ground" corresponds to K.

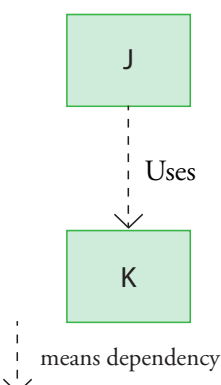
The class diagrams shown so far have omitted the detail of each class. This is fine if all that is of interest is the kind of relationship or association, e.g.



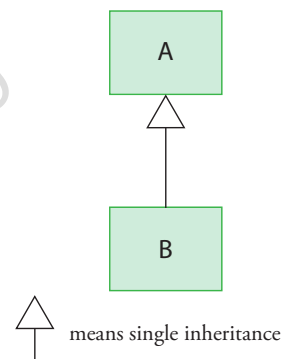
[Fig. 27.2 Shows how aggregation is represented in a class diagram](#)



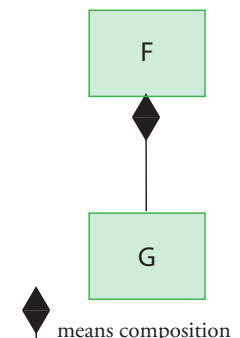
[Fig. 27.4 Shows how association is represented in a class diagram](#)



[Fig. 27.5 Shows how dependency is represented in a class diagram](#)



[Fig. 27.3 Shows how composition is represented in a class diagram](#)



[Fig. 27.1 Shows how single inheritance is represented in a class diagram](#)

Questions

- 1 What is the symbol used on a class diagram to indicate
 - (a) single inheritance
 - (b) composition
 - (c) aggregation
 - (d) association
 - (e) dependency?

single inheritance. However, sometimes a more informative and detailed class diagram is required. Each class in this more detailed class diagram would show details of its attributes and methods and their access level specifiers as shown [Figure 27.6](#). The bubble shows an example of the contents of a class rectangle. [Figure 27.7](#) shows the interpretation of the access level specifiers.

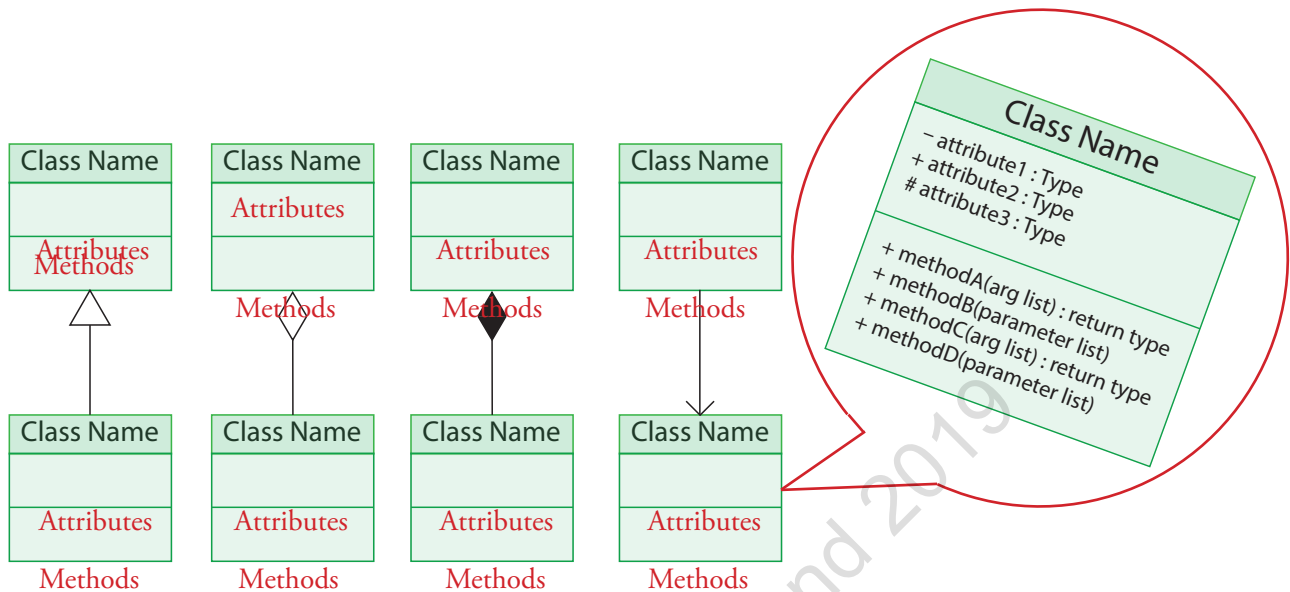


Fig. 27.6 Each class diagram is shown in more detail by splitting the rectangle into an attributes part and a methods part

Access level specifiers

+ means public - means private # means protected

Fig. 27.7 Access level specifiers for attributes and methods

Single inheritance class diagram

[Figure 27.8](#) shows an example based on an exercise used in [Chapter 23](#).

[Figure 27.9](#) class diagram is a more detailed version of [Figure 27.8](#). The attributes section is omitted in [Figure 27.9](#) from classes Duck and Crow because the exercise in [Chapter 23](#) did not supply these.

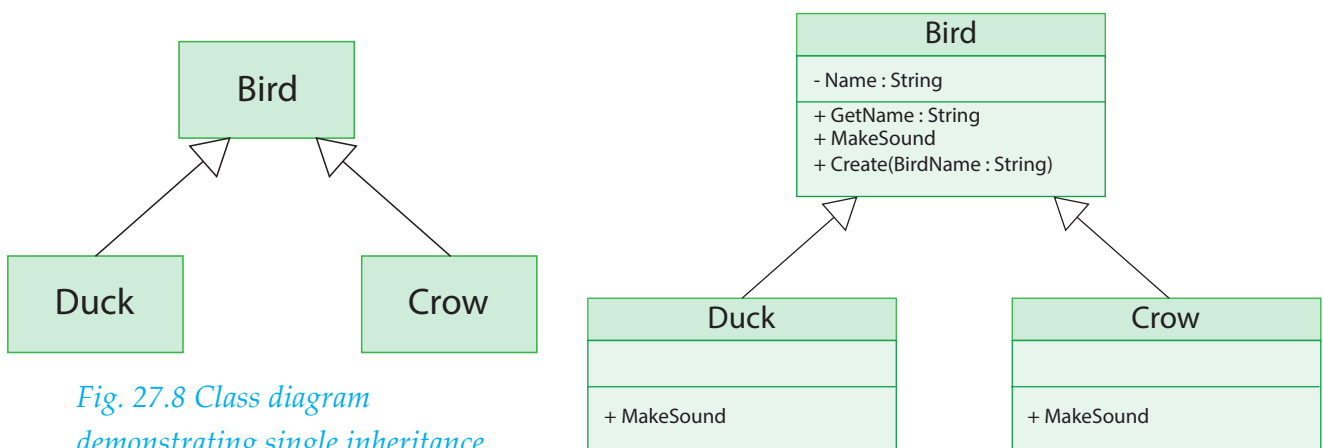


Fig. 27.8 Class diagram demonstrating single inheritance

Fig. 27.9 Class diagram demonstrating single inheritance and showing attributes and methods

Figure 27.9 shows the visibility levels of the attribute Name and the three methods GetName, MakeSound and Create. Visibility level is indicated by placing an access modifier or access/visibility level specifier, one of ‘-’, ‘+’, or ‘#’, before the attribute or method identifier.

If an attribute or method is private then the symbol used is ‘-’.

If an attribute or method is public then the symbol used is ‘+’.

If an attribute or method is protected then the symbol used is ‘#’.

Questions

- 2 Pythons and cobras are types of snake. Draw a single inheritance class diagram which shows the relationship between these three types if each is modelled as a class.

Composition class diagram

Figure 27.10 shows a composition class diagram for a noughts and crosses grid such as that shown in Figure 27.11. Composition on this class diagram is indicated by a filled diamond and line.

The degree of the association is shown as one grid is associated with nine squares by labelling the diamond with 1 and the other end of the composition symbol with 9. If the degree of association is variable then the 9-end is replaced by *n*.

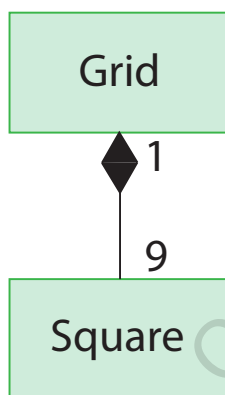


Fig. 27.10 Class diagram demonstrating composition

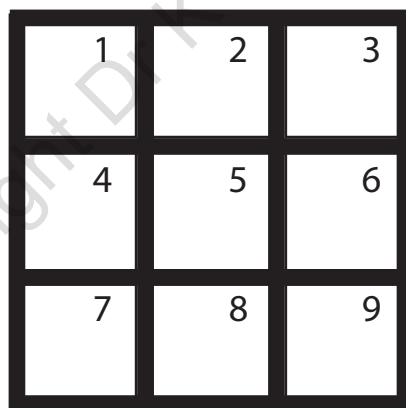


Fig. 27.11 Noughts and crosses grid

Key concept

Composition:

A whole/part relationship or association in which the aggregate object cannot exist without its components.

In composition, a composition hierarchy defines how an object is composed of other objects in a fixed relationship.

In composition, a composition hierarchy defines how an object is composed of other objects in a fixed relationship. The aggregate object cannot exist without its component objects. For example, a car comprises a body shell, an engine, a gearbox, seats, etc. A clock comprises a clock face, hands and clock mechanism. A car would not be a car without its components and a clock would not be a clock without its components. Therefore, we can say that composition is a **fixed whole/part relationship**, i.e. a special form of aggregation called **fixed aggregation**.

Questions

- 3 A snake has a fang. Draw a class diagram to show the relationship between snakes and fangs if each is modelled as a class.

Composition (fixed aggregation) is primarily a relationship between objects as is aggregation in general, rather than a relationship between classes. **A good test for a case of any form of aggregation is the "has a" test.** Describe the collection of objects first. If the phrase "has a" crops up between two objects then the first is composed of the second. For example, "a car has a body shell", "a car has a gearbox", etc. If none of the part objects are essential for the whole object to exist (i.e. the part objects can stay or go) then the association is just aggregation not the more restrictive form, composition.

Aggregation class diagram

Figure 27.12 shows a class diagram for aggregation. An unfilled diamond and line are used to indicate aggregation. The multiplicity of the relationship is indicated with '1' and '*' as shown and meaning one ('1') `TutorGroup` object is associated with many ('*') `Student` objects.

Aggregation is a whole/part association in which an object is composed of a variable set of component objects, e.g. an object of type `TutorGroup` can be associated with more than one object of type `Student` class, and different `Student` objects at different times. Similarly an object of type `SportsTeam` class can be associated with more than one object of type `Student` class, and different `Student` objects at different times. In each case, the objects chosen from the `Student` class are not always the same.

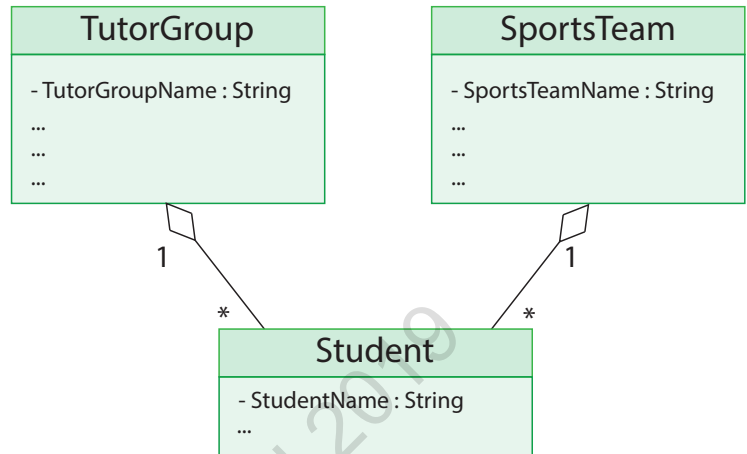


Fig. 27.12 Class diagram demonstrating aggregation

The whole can also exist without the part, e.g. a car boot object when empty.

To illustrate this consider the class diagram in Figure 27.13. This shows both composition and aggregation. The clock object cannot exist in the real world without its component objects, clock face, mechanism and hands but it can exist without its two batteries. These could be removed and used in another clock or device. Everyone knows that batteries have an independent existence.

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, and 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 object. The part is also capable of an independent existence, e.g. the contents of a car boot. It is possible for a part to be associated with more than one whole (not necessarily at the same time).

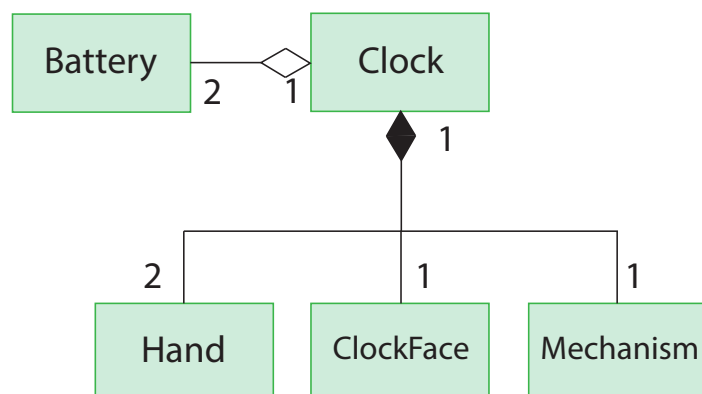


Fig. 27.13 Class diagram demonstrating fixed aggregation (composition) and variable aggregation (aggregation)

Fixed aggregation or composition

An object is composed of a fixed set of component objects, e.g. a rectangle is composed of four right-angle lines enclosing an area. The rectangle would not be a rectangle without all four lines - *Figure 27.14*.

Variable aggregation (aggregation)

An object is composed of a variable set of component objects, e.g. a car boot object can contain a variable number of objects and the ones used are not always the same - *Figure 27.15*. Multiplicity notation 0..* means zero or more.

Recursive or reflexive aggregation

The object contains components of its own type, like a Russian doll (each one containing a smaller one). For example, a rectangle object could contain two smaller rectangle objects - *Figure 27.16*.

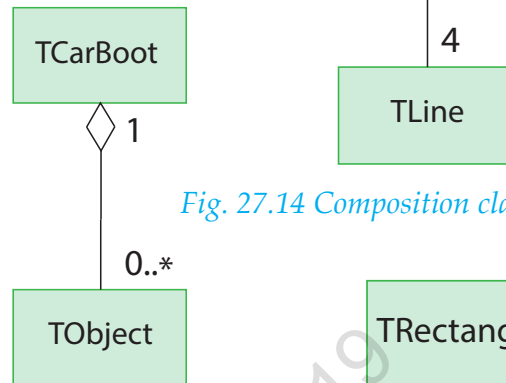


Fig. 27.14 Composition class diagram

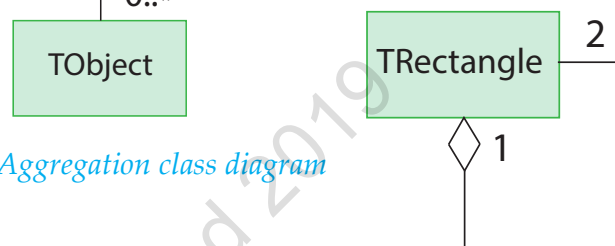


Fig. 27.15 Aggregation class diagram

Fig. 27.16 Recursive aggregation class diagram

Has/Has-a or is-a-part-of or contains relationship

Describe the collection of objects first. If the phrase "has a" or "has" or "is a part of" or "contains" crops up between two objects then the first is composed of the second or is an aggregate of the second.

Questions

- 4 A universal remote control is a part of home entertainment systems which consist of televisions, a cable TV set top box, blu-ray players and other electronic devices.
Draw a class diagram that shows the type of association between a universal remote controller and two types of electronic device, a TV and a blu-ray player if these are modelled as classes.
- 5 Draw a class diagram that shows the type of association between the wardrobe and its contents.
- 6 Draw a class diagram that shows the type of association between a bicycle object and the following objects: frame, handlebars, gears, wheels, brakes, removable lights.

Association

It is likely that you will encounter association which isn't a whole/part association (aggregation or composition).

Figure 27.17 shows an association class diagram for a Teacher class and a Student class. The association is one in which a teacher teaches a student. Clearly, this is not a whole/part association, i.e. a student is not a part of a teacher.

Contrast this with the example in Figure 27.12 in which a tutor group has many students and a sports team has many students.

The grid for noughts and crosses shown in Figure 27.11 does have an association relationship in addition to its aggregation one, it is a *uses* association: a noughts and crosses game *uses* a grid. This *uses* association is shown in Figure 27.18 and is a dependency association.

Bi-directional association also exists in which case the arrow has an arrowhead at each end.

In general, **association** is a type of relationship in which one object interacts with or uses another.

When the type is 'uses' we use the term **dependency** or **dependency association**. Dependency is a weaker variant of association between otherwise unrelated classes, such as between 'snakes' and 'long grass' – snakes use long grass to hide.

The most common usage of a dependency is to show that a method belonging to one class uses another class when an object of the one uses an object of another as a method parameter. For example, an application object, `Application`, of the class `TApplication` calls the method `Application.CreateForm` with the object parameter `Form1` of class `TForm1` as shown below and in Figure 27.19.

```
Application.CreateForm(TForm1, Form1);
```

Dependency also occurs when an object of one class initiates another object which is also the case in the example in Figure 27.19.

Dependency is shown as a dotted line with an arrowhead.



Fig. 27.17 Class diagram demonstrating association

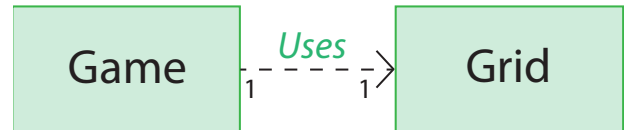


Fig. 27.18 Association class diagram for noughts and crosses game

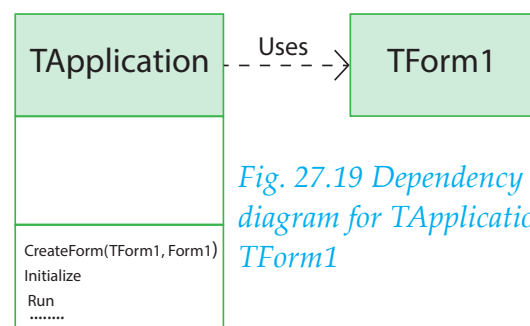


Fig. 27.19 Dependency class diagram for TApplication and TForm1

Questions

- 7 A doctor can be associated with multiple patients. At the same time, one patient can visit multiple doctors for treatment or consultation. Each of these objects has its own life cycle and there is no "owner" or parent. The objects that are part of the association relationship can be created and destroyed independently.

Draw a class diagram that shows the type of association between doctor and patient.

A class diagram shows the static relationships between classes. When two classes are connected by a relationship, the connection is called an association.

Associations may be classified as

- a dependency (uses)
- an aggregation
- a composition (a strong form of aggregation)
- inheritance (yes, even inheritance which is a generalisation/specialisation relationship or association)
- and if not any of the above then just an association

This is shown in [Figure 27.20](#).

Multiplicity

We can assign a **multiplicity** (also known as **arity**) to an association e.g. 1 and * as shown in [Figure 27.17](#).

Multiplicity specifies the number of objects in one class that relate to a single object of the associated class.

- 1 - one instance (exactly one) of the class, i.e. one object
- 0..1 - zero or one instances of the class, i.e. zero or one objects
- 0..* - zero or more instances of the class, i.e. zero or more objects
- 1..* - one or more instances of the class, i.e. one or more objects
- * - many instances of the class, i.e. many objects
- 0,3 - zero or three instances of the class, i.e. zero or three objects
- 5..10 - five through ten instances of the class, i.e. 5 or 6 or 7 or 8 or 9 or 10 objects.

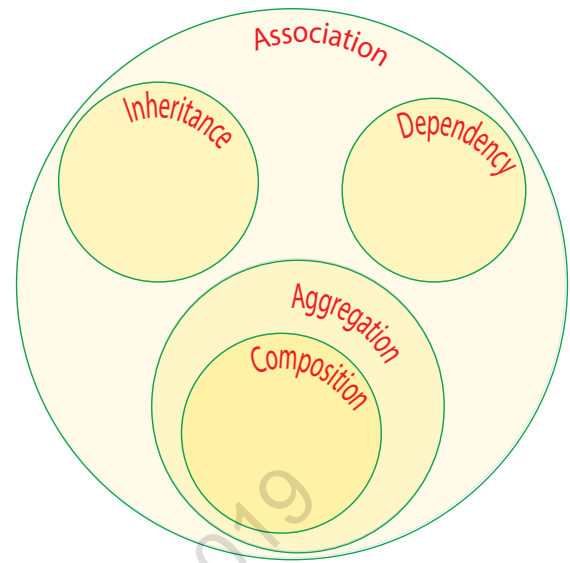
When no multiplicity is given, it is assumed to be 1.

Navigability

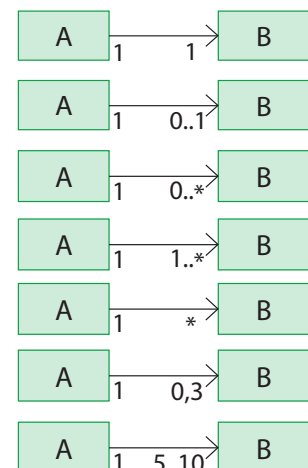
[Figure 27.17](#) expresses that `Teacher` must “know about” `Student`, but does `Student` need to know about `Teacher`?

This is known as **navigability**, i.e. how we can move from one object to another.

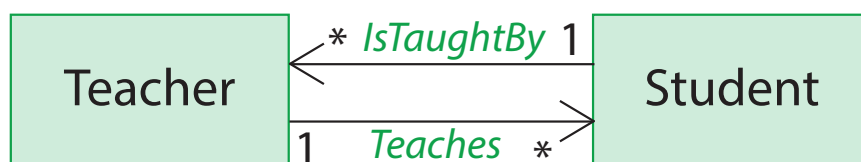
Navigability is represented in a class diagram by adding one or two arrows to the line representing an association or by using two single arrow lines as shown in [Figure 27.22](#).



[Fig. 27.20](#) Composition, Aggregation, Inheritance, Dependency are all forms of association



[Fig. 27.21](#) Multiplicity



[Fig. 27.22](#) Class diagram demonstrating two way association

To emphasise one-way navigation for composition or aggregation an arrow head can be added as shown in [Figure 27.23](#).

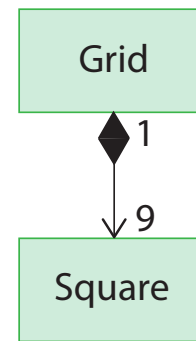


Fig. 27.23 Class diagram showing one way navigation

Object diagrams

Purpose: To learn how to create object diagrams

[Figure 27.24](#) shows an object diagram for three objects, `Person1`, `Dog1` and `Bird1` that have been created from the classes, `Person`, `Animal`, `Dog` and `Bird` shown in the class diagram [Figure 27.25](#).

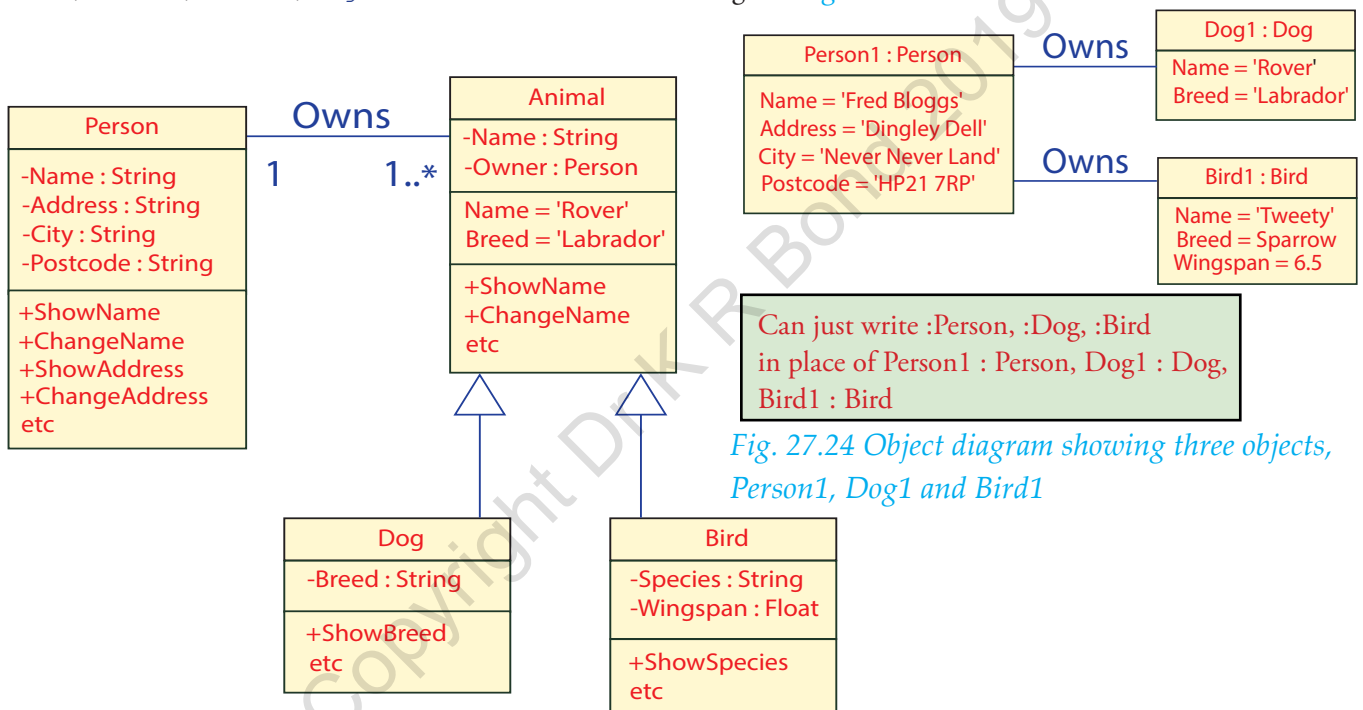


Fig. 27.24 Object diagram showing three objects, `Person1`, `Dog1` and `Bird1`

Fig. 27.25 Class diagram showing four classes, `Person`, `Animal`, `Dog` and `Bird`

Windows form application

[Figure 27.26](#) shows class and object diagrams for a windows form application with one button and three edit boxes.

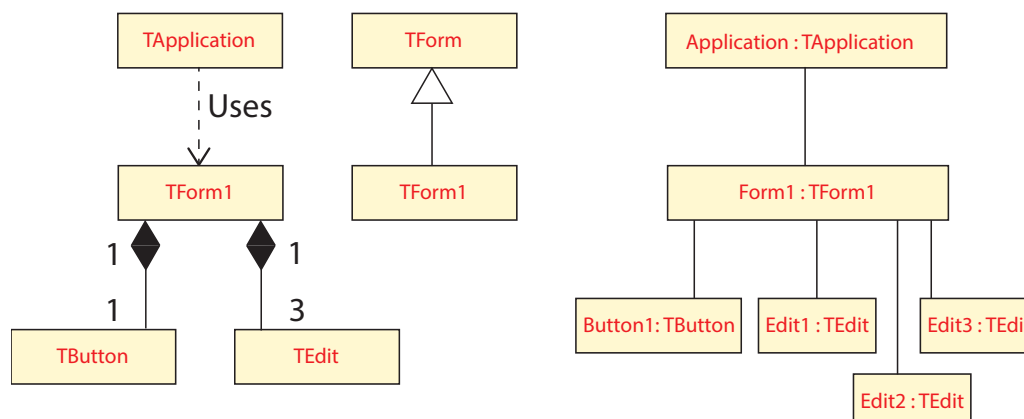


Fig. 27.26 Class and object diagrams for a windows form application consisting of five objects

Interface class diagram

Figure 27.27 shows an interface diagram for the diet exercise of [Chapter 26](#).

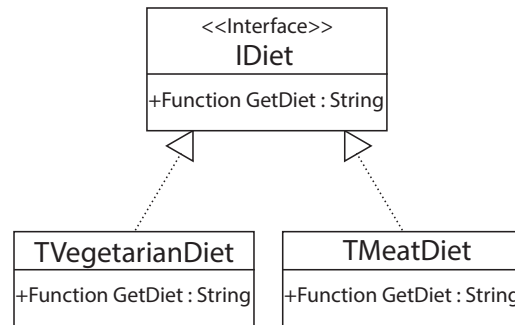


Fig. 27.27 Interface diagram for IDiet interface