Abstraction and Encapsulation

As discussed in Chapter 1, the essential idea behind the object-oriented approach to computer programming is that dividing a program into meaningful chunks, which can be written separately, offers lots of advantages. In the object-oriented approach to writing a program, chunks are chosen to represent the entities with which the program deals. The interesting behaviour of each type of entity is then defined by a corresponding class. When we come to write a new program, we first try to identify the types of entity with which the program deals. This allows us to decide what classes we will need to write. Subsequently, we have to decide exactly what behaviour each class should provide. In other words, we have to describe the essential properties of the entities that we are trying to represent and that are relevant to our program. Such a description is usually referred to as an abstraction. This process of designing meaningful abstractions of the entities of interest to a program is clearly at the heart of object-oriented programming.

Abstractions focus on the behaviour that the entities of interest exhibit rather than on how that behaviour is implemented. Clearly, writing a class whose instances actually exhibit the desired behaviour involves another level of complexity. Another essential feature of the object-oriented approach is that the users of a class shouldn’t have to be aware of that complexity but only of the behaviour defined by the abstraction that the class represents. The process of hiding this complexity is referred to as encapsulation.

Together abstraction and encapsulation allow us to divide up programs into chunks that are not only meaningful to the program at hand but are also largely independent of each other. As such, they are at the very heart of the object-oriented approach to computer programming, as we will discuss in the remainder of this chapter.
Abstraction

Let’s begin our discussion of abstraction by saying exactly what we mean by the term “abstraction”. Simply put, an abstraction is a description of the essential properties of an entity that are of interest. We often talk about “abstraction” or “building an abstraction of an entity” as the process of describing those essential properties of some entity.

A key part of building an abstraction of some entity is obviously identifying what the essential properties of interest are. Clearly, this depends on the type of entity that we are considering. However, it also depends on the context in which we are using the abstraction. Notice that our definition talks about the essential properties of interest. The phrase “of interest” implies that only a subset of the properties of an entity may need to be taken into account. The phrase “essential” implies that we have to be careful not to miss any properties of interest. In some sense, we must try to identify the smallest complete subset of the entity’s properties that are relevant to us. Why? Mainly, because we want to minimize complexity. By focusing on fewer properties we can hopefully simplify the abstraction and hence increase the ease with which it can be understood and, eventually, implemented.

As an example, let’s consider building an abstraction of a city. What are the essential properties of a city? Clearly, it depends on what our interest in the city is. From the perspective of a city planner, essential properties might include the population of the city, its demographic make up, the types and places of employment available, the availability of public utilities, such as water and power, and infrastructure, such as roads and public transportation. On the other hand, from the perspective of a visitor or potential visitor to the city, essential properties might include what places of interest there are to visit, what kinds of accommodation are available, what restaurants are available, and what kind of public transportation is available.

As you can see, building an abstraction is a very subjective process. There is no right abstraction of a particular entity, although there will certainly be abstractions that are more appropriate than others in different circumstances.

In general, an abstraction focuses on the behaviour of the entity in question rather than its implementation. In other words, an abstraction tells us what the entity does rather than how it does it. Put another way, it tells us how we might make use of the entity in question rather than how we might construct such as entity. For example, a meaningful abstraction of a car, from a driver’s perspective, would tell us how to drive the car, i.e., what controls are available, what the effect of each control is, and what we have to do to operate each control. It wouldn’t describe how the controls work or what components are necessary to make them work. We don’t need to know anything about how an engine is constructed in order to drive a car and, indeed, many drivers don’t! These details aren’t essential to the abstraction. It’s enough to know that to make the car go faster, we push the accelerator, while to make it go slower we push the brake pedal.

Abstraction in Object-oriented Programming

A key part of object-oriented programming is identifying the essential properties of the entities with which the program deals that are of interest. These essential properties are captured in the definition of a class whose instances represent those entities. As such, classes represent abstractions of the entities of interest to some program.

Since, building an abstraction is a subjective process, different classes might be required to represent the same type of entity in different circumstances. Thus, there is unlikely to be a single class that is appropriate for representing a given type of entity in every program.