Figure 4.8 *Class diagram showing the relationship between the classes Customer and Bike*

A class of objects is a group of objects with the same set of attributes, the same relationships and the same behaviour. When we study a problem domain, we will probably start by finding the objects in it and, once we have done that, work out what classes we need. In the Wheels system, we know we have 600 bike objects to represent. The next step is to agree upon a set of attributes those bikes have in common. Then we need to work out what we want our bike objects to be able to do. Once we have done that a Bike class can be defined with the agreed set of attributes, and operations that can deliver the required behaviour. We can then think and model in terms of classes of objects and their relationships to each other. To represent the hiring of Wheels' 600 bikes by 5000 customers, instead of having an object diagram with several thousand objects on it, we can summarise the information on a class diagram as shown in Figure 4.8. The notation is explained later in the chapter.

Although we start by identifying objects, it is the class that determines the structure and behaviour of its objects. We can compare a class to a scone cutter, or one of those animal-shaped cutters that children use to cut figures out of playdough. One cutter can be used to produce an infinite number of scones or animal figures. A class is used, like the scone cutter, as a template for creating objects in its form. Producing objects is a class's main role in life; a class is an object factory, it can produce hundreds of objects, all with exactly the same structure and behaviour. When we define a class we define the structure and the public interface for all objects of that class.

In object-oriented software, all objects belong to a class. Well-designed object-oriented code consists entirely of objects (and their classes); all of the system functionality is produced by the operations we define for the classes. In the software, the code that implements the operations is situated in the class. Objects of the class know about and access these operations, but don't carry round their own copy of them. For example, a Bike object knows that it can display its number and work out the cost of hiring it (or rather the bike it represents) for a given number of days. However, the code for these operations is part of the Bike class.

Objects are sometimes called instances of classes; the process of creating a new object belonging to a class is called *instantiation*.