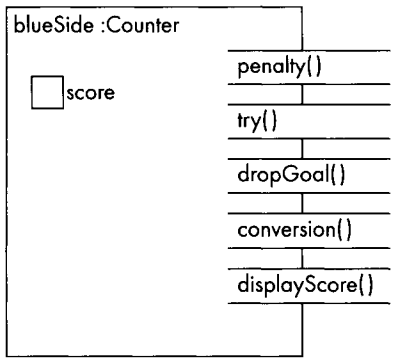

Figure 4.4 *A Counter object showing operations in its public interface*

conversion. If you are not familiar with the game of rugby don't worry about these terms. All we need to understand is that we can only increment the score attribute by 2, 3 or 5. In Figure 4.4 we have depicted the attribute score inside the protective wall of a rectangle representing the blueSide :Counter object.[2] The only way score can be updated is by using one of the operations we have defined for this object: penalty(), try(), dropGoal() or conversion(). This means that no other part of the program can inadvertently (or deliberately!) alter the score by using ordinary integer arithmetic, for example to subtract 10 or multiply by 4. The data is protected by the operations that encapsulate it. In fact the rest of the program need not even know that we have implemented it as an integer. All the rest of the program needs to know is the name of the object and the names of the operations it can use to correctly update or display blueSide's score. This is sometimes known as *data hiding.*

Public interface and messages. In the example in Figure 4.4 the operations penalty(), try(), dropGoal(), conversion() and displayScore() are what is known as the Counter object's public interface – all of these operations are defined as public, i.e. available for use by other parts of the program. Each operation has a *signature*, for example displayScore(), which forms the operation's public interface and must be used when the operation is invoked. The signature consists of the operation's name, parameter list, type of result and an indication of whether it is a private or public operation. An object's public interface is the only thing the rest of the program knows about the object apart from its name. The public interface provides the *services* it makes available to other objects. If another object wants to update blueSide's score – because the blue side have scored a try – it cannot do it directly, it

2. *Note that this is not a standard way of modelling objects, we only use it to illustrate encapsulation.*