Figure 15.2 What is *Polymorphism*?

Because a **Student** is a kind of **Person**, this is allowed:

Person p = new Student("Richard", "28/10/67", "890792492");

Question: What does

System.out.println(p.toString());

print? Is the **Person** or **Student** version of **toString()** invoked?

Answer: The **Student** version is invoked—name, birth date *and* student number are printed.

Java finds the method belonging to the actual object …

…and *not* the method belonging to the type specified for the variable (in this example, **p** is of type **Person**).

This is called *polymorphism*, because it allows a single variable to have different behaviours depending on the class of the object it actually refers to.

Sometimes you need a cast.

For example,

System.out.println(p.getStudentNumber());

fails because the compiler sees that **p** is a **Person**, and complains that **Person** has no **getStudentNumber()** method. Instead, use

System.out.println($\boxed{}$);

The cast tells the compiler that **p** actually is, or at least will be at run-time, a **Student**.