Figure 15.1 *Student* extends *Person*.

A **Student** is a kind of **Person**—but in addition to having a name and birth date, a **Student** also has a *student number*:

```
public class Student extends Person {
    private String stunum;    // this is the additional instance variable;
                              // the other two are inherited

    public Student (String name, String birthdate, String stunum) {
        [                                        ]
                              // initializes the inherited instance variables
        this.stunum = stunum;
    }

    public String getStudentNumber() {
        return stunum;
    }

    public String toString() {
        String result = [                    ]
                              // invokes the inherited toString()
        result += "\nStudent number: " + stunum;
        return result;
    }
}
```

Things to note about the **Student** class:

- You instantiate a **Student** object by calling a *three*-parameter constructor; for example

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

- A **Student** object *inherits* the **getName()** method from its parent class, **Person**, so you can use it without having to rewrite it in the **Student** class; for example

  System.out.println(stu.getName()); *// prints out Richard*

  This is *very* useful—especially if there are many inherited methods.

- We wrote a more informative **toString()** in the **Student** class by *overriding* **Person**'s **toString()**. Actually, we could have changed it completely, but instead we invoked **Person**'s **toString()** first and then added information to it.

  Question: Why is this strategy useful?