Figure 13.0 Overloading revisited.

In computing, to *overload* a name is to give it more than one meaning; which meaning is used depends on the context. Here are some examples:

“**+**” can mean either or , depending on whether its operands are or

System.out.println(5)	uses
System.out.println(true)	uses
System.out.println(5.2)	uses
System.out.println()	uses
System.out.println("hi")	uses

Which **println** is used is specified uniquely by its *signature*.

Let’s overload the **Student** class constructor again:

```
public class Student {
    private String name;
    private String stunum;
    private int credits;
    private double GPA;

    public Student(String name, String stunum) {
        this.name = name;
        this.stunum = stunum;
        this.credits = 0;
        this.GPA = 0.0;
    }

    public Student(String name, String stunum, int credits, double GPA) {
        this.name = name;
        this.stunum = stunum;
        this.credits = credits;
        this.GPA = GPA;
    }
}
```

This code is *repetitive*. It’s unpleasant to look at, and you are more likely to make errors when writing it. It’s much better to define the less-specific first constructor in terms of the more-specific second one:

```
    public Student(String name, String stunum) {

    }
```

this means “my constructor”; it can only be called like this in the first line of the calling constructor.