Figure 16.0 Object-orientedness—a review.

Let's review how our hierarchy of **Mark** classes illustrates the concepts of object-oriented programming:

1. Class extension and derived classes.

```
// base class or superclass
public class Mark {
    public int intValue () {
        return -9999;
    }
}

// child class / subclass / derived class
public class LetterGrade extends Mark {
    private String mark;

    public LetterGrade (String mark) {
        this.mark = mark;
    }

    public int intValue () {
        if (mark.equals("A+")) return 95;
        if (mark.equals("A")) return 88;
        …
    }
}

// another child class
public class PercentGrade extends Mark {
    private int mark;

    public PercentGrade (String mark) {
        this.mark = Integer.parseInt(mark);
    }

    public int intValue () {
        return mark;
    }
}
```

2. Method overriding.

   The **intValue()** methods in **PercentGrade** and **LetterGrade** *override* the definition in **Mark**. For example, if we have

   ```
   Mark m = new PercentGrade("75");
   ```

   then the call

   ```
   System.out.println(m.intValue());
   ```

   user **PercentGrade**'s definition of **intValue()**.