Figure 16.1 OO review, continued.

3. Polymorphism.

   “The quality of having many shapes”, or, in our context, “behaviours”.
   A single variable can behave differently, depending on its current value.

```
while (true) {
    String mark = keyb.readLine();
    if (mark.equals("quit")) break;
    System.out.print("letter grade? (y/n) ");
    Mark m;
    if (keyb.readLine().equals("y"))
        m = new LetterGrade(mark);
    else
        m = new PercentGrade(mark);
    System.out.println("You entered: " + m.intValue());
}
```

   The “behaviour” of the method call **m.intValue()** depends
   on the current value of **m**.

4. Inheritance.

   Sometimes the parent class’s definition is just fine. For example,
   let’s give the **Mark** class a method to calculate the grade point
   corresponding to the mark’s value:

```
public class Mark {
    public int intValue () {
        return -9999;
    }
    public double gradePoint () {
        int m = this.intValue();    // polymorphism used here!
        if (m >= 80) return 4.00;
        else if (m > 76) return 3.70;
        …
    }
}
```

   Now if we instantiate a **LetterGrade**,

```
LetterGrade mark = new LetterGrade("A-");
```

   we can print out its grade point value:

```
System.out.println(mark.gradePoint());
```

   The capability to calculate grade point values is *inherited*
   from the parent class.