where $f(x)$ is a function of x. The approximate solution as given by the Euler algorithm is

$$y_{n+1} = y_n + f(x_n)\Delta x. \tag{2.9}$$

Note that the rate of change of y has been approximated by its value at the *beginning* of the interval $f(x_n)$.

(a) Suppose that $f(x) = 2x$ and $y(x = 0) = 0$. The analytical solution is $y(x) = x^2$, which we can confirm by taking the derivative of $y(x)$. Convert (2.8) into a finite difference equation using the Euler algorithm. For simplicity, choose $\Delta x = 0.1$. It would be a good idea to first use a calculator or pencil and paper to determine y_n for the first several time steps.

(b) Sketch the difference between the exact solution and the approximate solution given by the Euler algorithm. What condition would the rate of change $f(x)$ have to satisfy for the Euler algorithm to give the exact answer? ■

Problem 2.2 Invent your own numerical algorithm

As we have mentioned, the Euler algorithm evaluates the rate of change of y by its value at the beginning of the interval $f(x_n)$. The choice of where to approximate the rate of change of y during the interval from x to $x + \Delta x$ is arbitrary, although we will learn that some choices are better than others. All that is required is that the finite difference equation must reduce to the original differential equation in the limit $\Delta x \to 0$. Think of several other algorithms that are consistent with this condition. ■

2.2 ■ SIMULATING FREE FALL

The source code for the *class* `FirstFallingBallApp` shown in Listing 2.1 is defined in a file named FirstFallingBallApp.java. The code consists of a sequence of *statements* that create *variables* and define *methods*. Each statement ends with a semicolon. Each source code file is compiled into *byte code* that can then be executed. The compiler places the byte code in a file with the same name as the Java source code file with the extension `class`. For example, the compiler converts FirstFallingBallApp.java into byte code and produces the FirstFallingBallApp.class file. One of the features of Java is that this byte code can be used by any computer that can run Java programs.

A Java application is a class that contains a *main* method. The following application is an implementation of the Euler algorithm given in (2.7). The program also compares the numerical and analytic results. We will next describe the syntax used in each line of the program.

Listing 2.1 First version of a simulation of a falling particle.

```
1 // example of a single-line comment statement (ignored by compiler)
2 package org.opensourcephysics.sip.ch02;      // location of file
3 // beginning of class definition
4 public class FirstFallingBallApp {
5    // beginning of method definition
6    public static void main(String[] args) {
7       // braces { } used to group statements
8       // indent statements within a block so that
9       // they can be easily identified
```



```
10      // following statements form the body of main method
11      // example of declaration and assignment statement
12      double y0 = 10;
13      double v0 = 0;     // initial velocity
14      double t = 0;      // time
15      double dt = 0.01; // time step
16      double y = y0;
17      double v = v0;
18      double g = 9.8;    // gravitational field
19      // beginning of loop, n++ equivalent to n = n + 1
20      for(int n = 0;n<100;n++) {
21         // repeat following three statements 100 times
22         y = y+v*dt; // indent statements in loop for clarity
23         v = v-g*dt; // use Euler algorithm
24         t = t+dt;
25      }                  // end of for loop
26      System.out.println("Results");
27      System.out.println("final time = "+t);
28      // display numerical result
29      System.out.println("y = "+y+" v = "+v);
30      // display analytic result
31      double yAnalytic = y0+v0*t-0.5*g*t*t;
32      double vAnalytic = v0-g*t;
33      System.out.println("analytic y = "+yAnalytic+" v = "+vAnalytic);
34   } // end of method definition
35 } // end of class definition
```

The first line in Listing 2.1 is an example of a single line comment statement. Comment statements are ignored by the computer but can be very important for the user. Multiple line comments begin with `/*` and end with `*/`. Javadoc comments begin with `/**` but have been removed from the code listings in the book to save space. Download the source code from the Open Source Physics website to view the complete code with documentation.

The second line in Listing 2.1 declares a *package* name, which corresponds to the location (directory) of the source and byte code files. According to the package declaration, the file FirstFallingBallApp.java is in the directory org/opensourcephysics/sip/ch02. The package statement must be the first noncomment statement in the source file. For organizational convenience, it is a good idea to put related files in the same package. When executing a Java program, the Java Virtual Machine (the run-time environment) will search a specific set of directories (called the classpath) for the relevant class files. The documentation for your local development environment will describe how to specify the classpath.

The fourth line in Listing 2.1 declares the class name, `FirstFallingBallApp`. The Java convention is to begin a class name with an uppercase letter. If a name consists of more than one word, the words are joined together, and each succeeding word begins with an uppercase letter (another Java convention). The keyword `public` means that this class can be used by any other Java class.

Braces are used to delimit a block of code. The left brace `{`, used after the name of the class, begins the body of the class definition, and the corresponding right brace `}`, inserted at the end of the code listing on line 35, ends the class definition.

The sixth line in Listing 2.1 begins the definition of the `main` method. A method describes a sequence of actions that use the associated data and can be *called* (invoked) within the class or by other classes. The `main` method has a special status in Java. To run a class as a stand-alone program (an application), the class must define the `main` method. (In contrast,