```
    public void reset() {
        control.setValue("Probability p of step to right", 0.5);
        control.setValue("Number of steps N", 100);
    }

    public static void main(String[] args) {
        SimulationControl.createApp(new WalkerApp());
    }
}
```

Problem 7.5 Random walks in one dimension

(a) In class `Walker` the steps are of unit length so that $a = 1$. Use `Walker` and `WalkerApp` to estimate the number of trials needed to obtain Δx^2 for $N = 20$ and $p = 1/2$ with an accuracy of approximately 5%. Compare your result for Δx^2 to the exact answer in (7.10). Approximately how many trials do you need to obtain the same relative accuracy for $N = 100$?

(b) Is $\langle x \rangle$ exactly zero in your simulations? Explain the difference between the analytical result and the results of your simulations. Note that we have used the same notation $\langle \ldots \rangle$ to denote the exact average calculated analytically and the approximate average computed by averaging over many trials. The distinction between the two averages should be clear from the context.

(c) How do your results for $\langle x \rangle$ and Δx^2 change for $p \neq q$? Choose $p = 0.7$ and determine the N dependence of $\langle x \rangle$ and Δx^2.

*(d) Determine Δx^2 for $N = 1$ to $N = 5$ by enumerating all the possible walks. For simplicity, choose $p = 1/2$ so that $\langle x \rangle = 0$. For $N = 1$ there are two possible walks: one step to the right and one step to the left. In both cases $x^2 = 1$, and hence $\langle x_1{}^2 \rangle = 1$. For $N = 2$ there are four possible walks with the same probability: (i) two steps to the right, (ii) two steps to the left, (iii) first step to the right and second step to the left, and (iv) first step to the left and second step to the right. The value of x_2^2 for these walks is 4, 4, 0, and 0, respectively, and hence $\langle x_2^2 \rangle = (4+4+0+0)/4 = 2$. Write a program that enumerates all the possible walks of a given number of steps and compute the various averages of interest exactly. ■

The class `WalkerApp` displays the distribution of values of the displacement x after N steps. One way of determining the number of times that the variable x has a certain value would be to define a one-dimensional array, `probability`, and let

```
probability[x] += 1;
```

In this case because x takes only integer values, the array index of `probability` is the same as x itself. However, the above statement does not work in Java because x can be negative as well as positive. What we need is a way of mapping the value x to a bin or index number. The `HistogramFrame` class, which is part of the Open Source Physics display package, does this mapping automatically using the Java `Hashtable` class. In simple data structures, data is accessed by an index that indicates the location of the data in the data structure. Hashtable data is accessed by a *key*, which in our case is the value of x. A hashing function converts the key to an index. The `append` method of the `HistogramFrame` class takes a value, finds the index using a hashing function, and then increments the data associated with that key. The `HistogramFrame` class also draws itself.

The `HistogramFrame` class is very useful for taking a quick look at the distribution of values in a data set. You do not need to know how to group the data into bins or the range of values of the data. The default bin width is unity, but the bin width can be set using the `setBinWidth` method. See `WalkerApp` for an example of the use of the `HistogramFrame` class. Frequently, we wish to use the histogram data to compute other quantities. You can collect the data using the Data Table menu item in `HistogramFrame` and copy the data to a file. Another option is to include additional code in your program to analyze the data. The following statements assume that a `HistogramFrame` object called `histogram` has been created and data entered into it.

```
// creates array entries of data from histogram
java.util.Map.Entry[] entries = histogram.entries();
for (int i = 0, length = entries.length; i < length; i++) {
    // gets bin number
    Integer binNumber = (Integer) entries[i].getKey();
    // gets number of occurrences for bin number i
    Double occurrences = (Double) entries[i].getValue();
    // gets value of left edge of bin
    double value =
          histogram.getLeftMostBinPosition(binNumber.intValue());
    // sets value to middle of bin
    value += 0.5*histogram.getBinWidth();
    // convert from Double class to double data type
    double number = occurrences.doubleValue();
    // use value and number in your analysis
}
```

Problem 7.6 Nature of the probability distribution

(a) Compute $P_N(x)$, the probability that the displacement of the walker from the origin is x after N steps. What is the difference between the histogram, that is, the number of occurrences, and the probability? Consider $N = 10$ and $N = 40$ and at least 1000 trials. Does the qualitative form of $P_N(x)$ change as the number of trials increases? What is the approximate width of $P_N(x)$ and the value of $P_N(x)$ at its maximum for each value of N?

(b) What is the approximate shape of the envelope of $P_N(x)$? Does the shape change as N is increased?

(c) Fit the envelope $P_N(x)$ for sufficiently large N to the continuous function

$$C \frac{1}{\sqrt{2\pi\,\Delta x^2}} e^{-(x-\langle x \rangle)^2/2\Delta x^2}. \tag{7.11}$$

The form of (7.11) is the standard form of the Gaussian distribution with $C = 1$. The easiest way to do this fit is to plot your results for $P_N(x)$ and the form (7.11) on the same graph using your results for $\langle x \rangle$ and Δx^2 as input parameters. Visually choose the constant C to obtain a reasonable fit. What are the possible values of x for a given value of N? What is the minimum difference between these values? How does this difference compare to your value for C? ■