Section 2.9 starts below…

2.9 Classes for representing geometric points

In this section we illustrate the use of the mathematical class library in Java. We also illustrate how a seemingly simple problem can be solved in several rather different ways. You will have the chance to analyze the advantages and disadvantages of various alternatives.

The classes described in this section represent points on a 2-dimensional plane. From mathematics, we know that to represent a point on a plane, you can use x and y coordinates, which are called *Cartesian coordinates*. Alternatively, you can use *polar coordinates*, represented by a radius (often called rho) and an angle (often called theta). In the code we have provided, you can interchangeably work with a given point as Cartesian coordinates or polar coordinates.

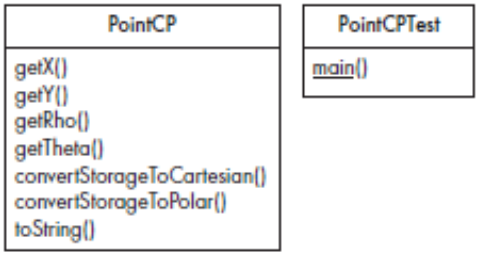

Figure 2.11 **Classes for representing points using both Cartesian and polar coordinates. Only the operations are shown**

Java already has classes for representing geometric points. Take a few moments to look at classes `Point2D` and `Point` in the Java documentation.

We will call the point class presented here `PointCP`; its main distinguishing feature from the built-in Java classes is that it can handle both Cartesian and polar coordinates. We also provide a class called `PointCPTest` which, like `PostalTest`, simply provides a user interface for testing. The public methods of both classes are shown in Figure 2.11. The code for these classes can also be found at the book's web site (www.lloseng.com).

Class `PointCP` contains two private instance variables that can either store x and y, or else rho and theta. No matter which storage format is used, all four possible parameters can be computed. Users of the class can also call methods `convertStorageToPolar` or `convertStorageToCartesian` in order to explicitly convert the internal storage of an instance to the alternative format.

The above design of `PointCP` is certainly not the only possible design. Table 2.1 shows several alternative designs; the above design is Design 1.

Exercises

E25 Answer the following questions with respect to the above designs of the `PointCP` class.

(a) Discuss why it might be useful to allow users of class `PointCP` (Design 1) to explicitly change the internal storage format, using `convertStorageToCartesian` or `convertStorageToPolar`.

(b) What might be a potential hidden weakness of these methods? Hint: what could happen if one is called, then the other, and this process is repeated numerous times?

(c) Write a short program to test whether the weakness you discussed in part b is, in fact, real.

E26 Create a table describing the various advantages (pros) and disadvantages (cons) of each of the five design alternatives. Some of the factors to consider are: simplicity of code, efficiency when creating instances, efficiency when

Table 2.1 Alternative designs for the `PointCP` class

	How Cartesian coordinates are computed	*How polar coordinates are computed*
Design 1: Store one type of coordinates using a single pair of instance variables, with a flag indicating which type is stored	Simply returned if Cartesian is the storage format, otherwise computed	Simply returned if polar is the storage format, otherwise computed
Design 2: Store polar coordinates only	Computed on demand, but not stored	Simply returned
Design 3: Store Cartesian coordinates only	Simply returned	Computed on demand, but not stored
Design 4: Store both types of coordinates, using four instance variables	Simply returned	Simply returned
Design 5: Abstract superclass with designs 2 and 3 as subclasses	Depends on the concrete class used	Depends on the concrete class used

doing computations that require both coordinate systems, and amount of memory used.

E27 Implement and test Design 5. You will also have to make some small changes to `PointCPTest`. Hints: a) Do you still need the variable `typeCoord`? b) Do still you need the third argument in the constructor?

E28 Run a performance analysis in which you compare the performance of Design 5, as you implemented it in the previous exercise, with Design 1. Determine the magnitude of the differences in efficiency, and verify the hypotheses you developed in E26.

E29 To run a performance analysis, you will have to create a new test class that randomly generates large numbers of instances of `PointCP`, and performs operations on them, such as retrieving polar and Cartesian coordinates. You should then run this test class with the two versions of `PointCP` – Design 1 and Design 5.

E30 Summarize your results in a table: the columns of the table would be the two designs; the rows of the table would be the operations. The values reported in the table would be the average computation speed. Make sure you explain your results.