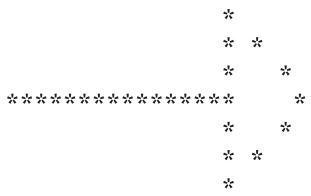
PROGRAMMING PROJECTS

Programming Projects require more problem-solving than Practice Programs and can usually be solved many different ways. Visit www.myprogramminglab.com to complete many of these Programming Projects online and get instant feedback.

1. Define a class called `Diamond` that is derived from either the class `ShapeBasics` (Listing 8.12) or the abstract class `ShapeBase` (Listing 8.19). A diamond has the same sort of top half as a `Triangle` object, and its bottom half is an inverted version of its top half. Define a utility class having public static methods, such as the method `skipSpaces` and other methods that draw horizontal lines, big V's, and inverted big V's. Recall that Self-Test Question 31 asked you to describe one method in this class.

2. Define two derived classes of the abstract class `ShapeBase` in Listing 8.19. Your two classes will be called `RightArrow` and `LeftArrow`. These classes will be like the classes `Rectangle` and `Triangle`, but they will draw arrows that point right and left, respectively. For example, the following arrow points to the right:



   The size of the arrow is determined by two numbers, one for the length of the "tail" and one for the width of the arrowhead. (The width is the length of the vertical base.) The arrow shown here has a length of 16 and a width of 7. The width of the arrowhead cannot be an even number, so your constructors and mutator methods should check to make sure that it is always odd. Write a test program for each class that tests all the methods in the class. You can assume that the width of the base of the arrowhead is at least 3.

3. Define two classes, `Patient` and `Billing`, whose objects are records for a clinic. Derive `Patient` from the class `Person` given in Listing 8.1. A `Patient` record has the patient's name (defined in the class `Person`) and identification number (use the type `String`). A `Billing` object will contain a `Patient` object and a `Doctor` object (from Practice program 2). Give your classes a reasonable complement of constructors and accessor methods, and an `equals` method as well. First write a driver program to test all your methods, then write a test program that creates at least two patients, at least two doctors, and at least two `Billing` records and then displays the total income from the `Billing` records.